STC89C51RC / RD+ 系列单片机器件手册

---加密性强

---低功耗,超低价

--- 高速，高可靠

--- 强抗静电，强抗干扰

STC89C51RC,　STC89LE51RC
STC89C52RC,　STC89LE52RC
STC89C53RC,　STC89LE53RC

STC89C54RD+, STC89LE54RD+
STC89C58RD+, STC89LE58RD+
STC89C516RD+, STC89LE516RD+

STC89LE516AD, STC89LE58AD
STC89LE54AD, STC89LE52AD

STC12C5404AD, STC12C5404
STC12C2052AD, STC12C2052

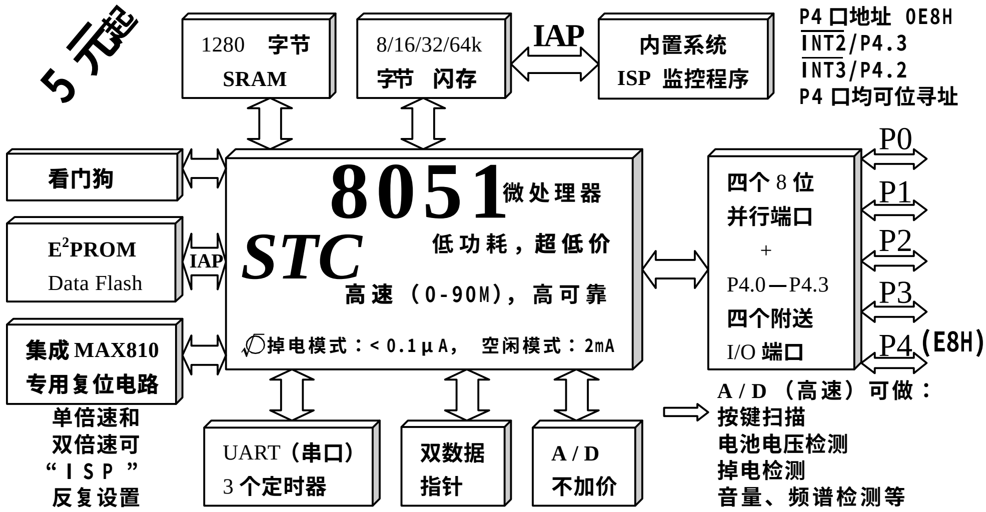
推荐优先选择代表宏晶最新水平的 STC12C54xx 系列单片机

宏晶科技

www.MCU-Memory.com

Update date: 2007-11-17

STC 89 系列单片机，高速、高可靠、在线编程
—— 提升的是性能，降低的是成本



选择 STC89C52RC 系列
STC89C58RD+ 系列单片机的理由：

★加密性强

★超强抗干扰：
1、高抗静电（ESD 保护）
2、轻松过 2KV/4KV 快速脉冲干扰(EFT 测试)
3、宽电压，不怕电源抖动
4、宽温度范围，-40℃～85℃

★三大降低单片机时钟对外部电磁辐射的措施：
—— 出口欧美的有力保证
1、禁止 ALE 输出；
2、如选 6 时钟/机器周期，外部时钟频率可降一半；
3、单片机时钟振荡器增益可设为 1/2gain。

★超低功耗：
1、掉电模式：　典型功耗　＜0.1μA
2、正常工作模式：　典型功耗　4mA - 7mA
3、掉电模式可由外部中断唤醒，适用于电池供电系统，如水表、气表、便携设备等。

★在系统可编程，无需编程器，无需仿真器
★可送 STC-ISP 下载编程器，1 万片/人/天
★可供应内部集成 MAX810 专用复位电路的单片机，只有 D 版本才有内部集成专用复位电路，原复位电路可以保留，也可以不用，不用时 RESET 脚接 1K 电阻到地

STC 89 系列单片机选型一览表　超低价

型　号	最高时钟频率 Hz 5V	最高时钟频率 Hz 3V	Flash 存储器	RAM 字节	降低 EMI	看门狗	双倍速	P4 口	ISP	IAP	E2P ROM 字节	A/D
STC 89C51 RC	0～80M		4K	512	√	∅	√	√	√	√	2K+	
STC 89C52 RC	0～80M		8K	512	√	∅	√	√	√	√	2K+	
STC 89C53 RC	0～80M		15K	512	√	∅	√	√	√	√		
STC 89C54 RD+	0～80M		16K	1280	√	∅	√	√	√	√	16K+	
STC 89C55 RD+	0～80M		20K	1280	√	∅	√	√	√	√	16K+	
STC 89C58 RD+	0～80M		32K	1280	√	∅	√	√	√	√	16K+	
STC 89C516 RD+	0～80M		64K	1280	√	∅	√	√	√	√		
STC 89LE51 RC		0～80M	4K	512	√	∅	√	√	√	√	2K+	
STC 89LE52 RC		0～80M	8K	512	√	∅	√	√	√	√	2K+	
STC 89LE53 RC		0～80M	15K	512	√	∅	√	√	√	√		
STC 89LE54 RD+		0～80M	16K	1280	√	∅	√	√	√	√	16K+	
STC 89LE58 RD+		0～80M	32K	1280	√	∅	√	√	√	√	16K+	
STC 89LE516RD+		0～80M	64K	1280	√	∅	√	√	√	√		
STC 89LE516AD	0～90M, 3.6～2.4V		64K	512	√		√	√	√			√

关于单片机说明：<管脚与流行的 8051 兼容>　　大客户超低价

DIP-40，PLCC-44，LQFP-44 封装(RC/RD+ 系列 PLCC、LQFP 有 P4 口地址 E8H，AD 系列 P4 口为 C0H)

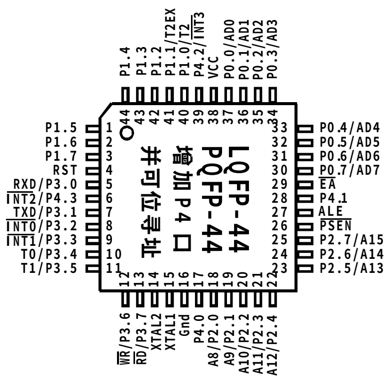
RC/RD+ 系列 PLCC、LQFP 多两个外部中断 P4.2/INT3，P4.3/INT2。 P4 口均可位寻址

5V: 5.5V～3.8V；3V: 3.8V～2.4V(仅针对 RC/RD+ 系列)

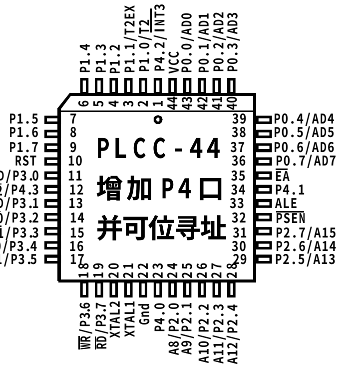
∅ 真正的看门狗，可放心省去外部看门狗，缺省为关闭，打开后无法关闭。单倍速和双倍速可反复设置

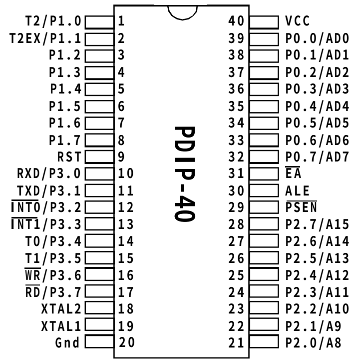
∅ “6 时钟/机器周期”和“12 时钟/机器周期”可在 ISP 编程时反复设置，新的设置冷启动后才生效

★强烈推荐 LQFP44 小型封装★

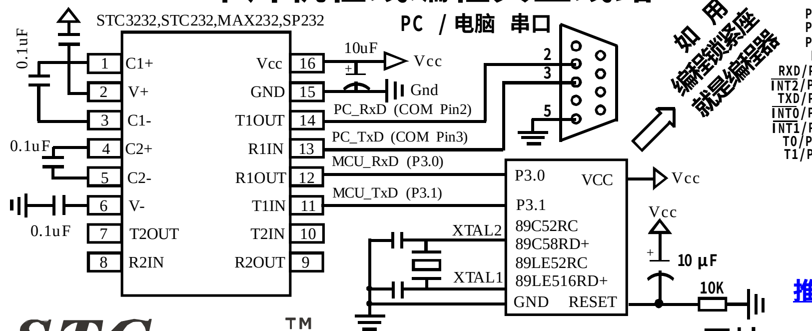


★尽量不选落后的 PDIP 和 PLCC 封装★





STC　单片机在线编程典型线路



推荐优先选择代表宏晶最新水平的 STC12C5404AD 系列单片机

STC micro™
宏　晶　科　技
8051 单片机全球第一品牌
中国本土 MCU 领航者
新客户请直接联系深圳以获得更好的技术支持和服务

网址：www.MCU-Memory.com　技术支持：13922805190

深　圳：Tel：0755-82948411　82948412　Fax：0755-82944243　82905966
广州办：Tel：020-87501705　85518657　Fax：020-85517881
上海办：Tel：021-53560136　53560138　Fax：021-53080587
北京办：Tel：010-62538687　62634001　Fax：010-62538683

免费索取
从网上下载样品申请单，
传真至深圳申请 STC 单片机
样片及 ISP 下载线/编程工具

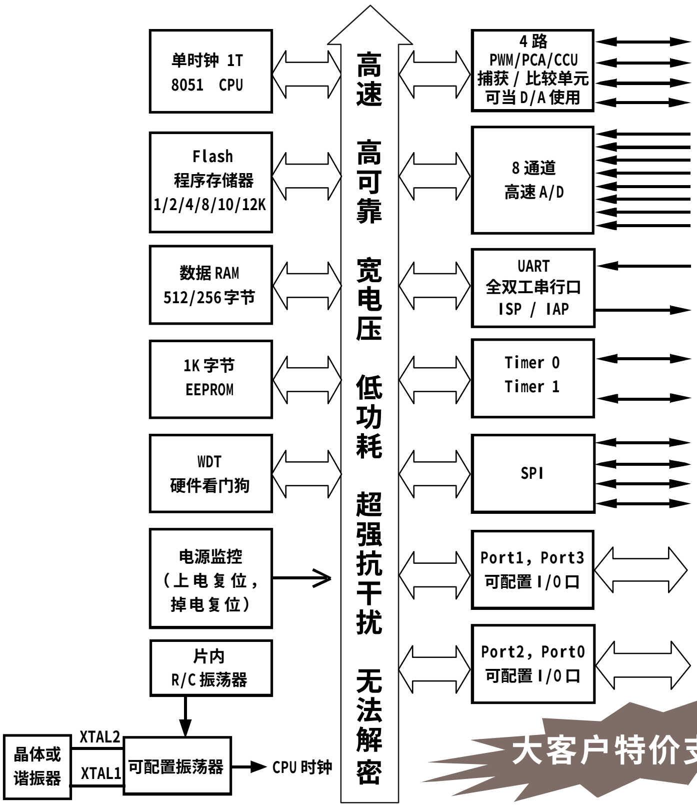
领导业界革命 覆盖市场需求

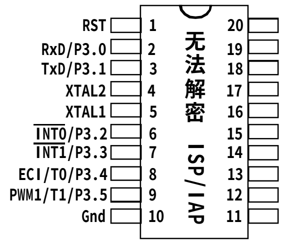
STC 12C5410AD 系列 1T 8051 单片机

——1 个时钟 / 机器周期，高速、高可靠，4 路 PWM，8 路高速 A/D 转换

宏晶科技是新一代增强型 8051 单片机标准的制定者和领导厂商，致力于提供满足中国市场需求的世界级高性能单片机技术，在业内处于领先地位，销售网络覆盖全国。在高品质的基础上，以极低的价格和完善的服务赢得了客户的长期信赖。在广受欢迎的 STC89C51 全系列单片机的基础上，现全力推出 “1 个时钟 / 机器周期” 的单片机，全面提升 8051 单片机性能。欢迎海内外厂家前来洽谈合作！新客户请直接联系深圳，以获得更好的技术支持与服务。

强烈推荐选择 SOP-20/28/32 贴片封装，尽量不选落后的插件 DIP 封装

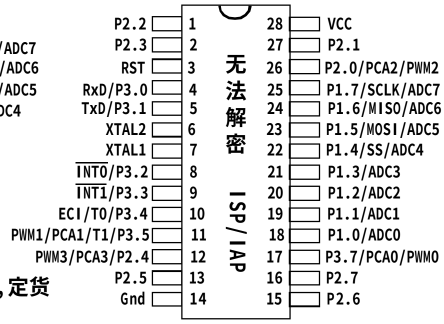


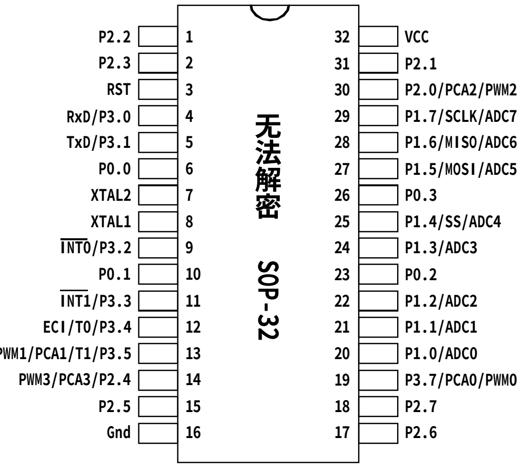


SOP-20, DIP-20

超小封装 TSSOP-20,6.4mm X 6.4mm,定货

SOP-28 / SKDIP-28 (窄体)





STC12C5410/STC12C2052 系列主要性能：

- 高速：1 个时钟 / 机器周期，增强型 8051 内核，速度比普通 8051 快 8～12 倍
- 宽电压：5.5～3.8V，2.4～3.8V（STC12LE5410AD 系列）
- 低功耗设计：空闲模式，掉电模式（可由外部中断唤醒）
- 工作频率：0～35MHz，相当于普通 8051：0～420MHz
- 时钟：外部晶体或内部 RC 振荡器可选，在 ISP 下载编程用户程序时设置
- 12K/10K/8K/6K/4K/2K 字节片内 Flash 程序存储器，擦写次数 10 万次以上
- 512 字节片内 RAM 数据存储器
- 芯片内 EEPROM 功能
- ISP / IAP，在系统可编程 / 在应用可编程，无需编程器 / 仿真器
- 10 位 ADC，8 通道，STC12C2052AD 系列为 8 位 ADC。4 路 PWM 还可当 4 路 D/A 使用
- 4 通道捕获 / 比较单元（PWM/PCA/CCU），STC12C2052AD 系列为 2 通道
  --- 也可用来再实现 4 个定时器或 4 个外部中断（支持上升沿 / 下降沿中断）
- 2 个硬件 16 位定时器，兼容普通 8051 的定时器。4 路 PCA 还可再实现 4 个定时器
- 硬件看门狗（WDT）
- 高速 SPI 通信端口
- 全双工异步串行口(UART)，兼容普通 8051 的串口
- 先进的指令集结构，兼容普通 8051 指令集
  4 组 8 个 8 位通用工作寄存器（共 32 个通用寄存器）
  有硬件乘法 / 除法指令
- 通用 I/O 口（27/23/15 个），复位后为： 准双向口 / 弱上拉（普通 8051 传统 I/O 口）
  可设置成四种模式：准双向口 / 弱上拉，推挽 / 强上拉，仅为输入 / 高阻，开漏
  每个 I/O 口驱动能力均可达到 20mA，但整个芯片最大不得超过 55mA

STC12C1052 ，人民币 3.8 元

如选 32-Pin，推荐选 SOP-32，PLCC-32 不再生产

选择 STC12C5410AD 系列单片机的理由：

★ 加密性强，无法解密

★ 超强抗干扰：
1、高抗静电（ESD 保护）
2、轻松过 4KV 快速脉冲干扰(EFT 测试)
3、宽电压，不怕电源抖动
4、宽温度范围，-40℃～85℃

★ 1 个时钟 / 机器周期，可用低频晶振，大幅降低 EMI
--- 出口欧美的有力保证

★ 超低功耗：
1、掉电模式： 典型功耗 <0.1 μA
2、空闲模式： 典型功耗 2mA
3、正常工作模式： 典型功耗 2.7mA - 7mA
4、掉电模式可由外部中断唤醒，适用于电池供电系统，如水表、气表、便携设备等。

★ 在系统可编程，无需编程器，无需仿真器，可远程升级

★ 可送 STC-ISP 下载编程器，1 万片 / 人 / 天

★ 内部集成 MAX810 专用复位电路，原复位电路可以保留，也可以不用，不用时 RESET 脚接 1K 电阻到地。

STC micro™
宏 晶 科 技
8051 单片机全球第一品牌
中国本土 MCU 领航者
新客户请直接联系深圳以获得更好的技术支持和服务

网址：www.MCU-Memory.com 技术支持：13922805190

深 圳：Tel：0755-82948411 82948412 Fax：0755-82944243 82905966
广州办：Tel：020-87501705 85518657 Fax：020-85517881
上海办：Tel：021-53560136 53560138 Fax：021-53080587
北京办：Tel：010-62538687 62634001 Fax：010-62538683

免费索取

从网上下载样品申请单，
传真至深圳申请 STC 单片机
样片及 ISP 下载线 / 编程工具

STC12C5410AD 系列 1T 8051 单片机选型一览表

型 号	工作电压(V)	Flash程序存储器字节	SRAM字节	定时器	UART	PCA 16位PWM 8位	A/D 8路	I/O	看门狗	内置复位	EEP ROM	SPI	封装20-Pin	封装28-Pin	封装32-Pin
STC12C2052AD系列单片机选型一览															
STC12C1052	5.5 - 3.5	1K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP	管脚兼容 89C2051 超强抗干扰 无法解密	
STC12C1052AD	5.5 - 3.5	1K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12C2052	5.5 - 3.5	2K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP		
STC12C2052AD	5.5 - 3.5	2K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12C4052	5.5 - 3.5	4K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP		
STC12C4052AD	5.5 - 3.5	4K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12C5052	5.5 - 3.5	5K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP		
STC12C5052AD	5.5 - 3.5	5K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE1052	2.2 - 3.8	1K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE1052AD	2.2 - 3.8	1K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE2052	2.2 - 3.8	2K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE2052AD	2.2 - 3.8	2K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE4052	2.2 - 3.8	4K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE4052AD	2.2 - 3.8	4K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE5052	2.2 - 3.8	5K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE5052AD	2.2 - 3.8	5K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12C5410AD系列单片机选型一览															
STC12C5402	5.5 - 3.5	2K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5402AD	5.5 - 3.5	2K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5404	5.5 - 3.5	4K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5404AD	5.5 - 3.5	4K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5406	5.5 - 3.5	6K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5406AD	5.5 - 3.5	6K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5408	5.5 - 3.5	8K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5408AD	5.5 - 3.5	8K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5410	5.5 - 3.5	10K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5410AD	5.5 - 3.5	10K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5412	5.5 - 3.5	12K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5412AD	5.5 - 3.5	12K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5402	2.2 - 3.8	2K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5402AD	2.2 - 3.8	2K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5404	2.2 - 3.8	4K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5404AD	2.2 - 3.8	4K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5406	2.2 - 3.8	6K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5406AD	2.2 - 3.8	6K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5408	2.2 - 3.8	8K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5408AD	2.2 - 3.8	8K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5410	2.2 - 3.8	10K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5410AD	2.2 - 3.8	10K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5412	2.2 - 3.8	12K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5412AD	2.2 - 3.8	12K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP

目录

第 1 章　STC 单片机宣传资料 2
1.1 STC89 系列单片机宣传资料 2
1.2 STC12 系列单片机宣传资料 3
第 2 章　STC 单片机总体介绍 7
2.1 STC89C51RC/RD+系列单片机简介 7
2.2 STC89C51RC/RD+系列单片机选型一览表 8
2.3 STC89C51RC/RD+系列单片机管脚图及封装尺寸图 9
2.3.1 STC89C51RC/RD+系列单片机管脚图 9
2.3.2 STC89C51RC/RD+系列单片机封装尺寸图 10
2.4 STC89C51RC/RD+系列单片机命名规则 14
2.5 STC89C51RC/RD+系列单片机优点及特性 15
2.6 STC89C51RC/RD+系列单片机典型应用电路 16
2.6.1 STC89C51RC/RD+系列单片机 D 版本典型应用电路（现大批量供货的产品） 16
2.7 STC89C51RC/RD+系列单片机特殊功能寄存器映像 说明 SFR Mapping 17
2.8 STC89C51RC/RD+系列单片机中断系统 20
2.9 降低单片机时钟对外界的电磁辐射（EMI）--- 三大措施 21
2.10 STC89C51RC/RD+系列单片机内部扩展 RAM 的使用 / 禁止 22
2.11 STC89C51RC/RD+系列单片机双数据指针 DPTR0，DPTR1 的使用 29
2.12 STC89C51RC/RD+系列单片机扩展 P4 口的使用（可以位寻址） 30
第 3 章　STC 单片机的看门狗及软件复位 31
3.1 看门狗应用及测试程序 31
3.1.1 看门狗应用介绍 31
3.1.2 一个完整的看门狗测试程序，在下载板上可以直接测试 33
3.2 如何用软件实现系统复位 34
第 4 章　STC89C51RC/RD+系列单片机 IAP 及 EEPROM 应用说明 35
4.1 IAP 及 EEPROM 应用 35
4.2 IAP/EEPROM 汇编简介 40
4.3 一个完整的 IAP/EEPROM 测试程序，在下载板上可以直接测试 43
第 5 章　STC89C51RC/RD+系列单片机定时器的使用及测试程序 47
5.1 定时器 0/1 的介绍及测试程序 47
5.1.1 定时器 0/1 的介绍 47
5.1.2 定时器 0/1 应用程序举例 51
5.1.3 用定时器 1 做波特率发生器（一个完整的程序，在下载板上可以直接测试 56
5.2 定时器 2 的介绍 63
5.2.1 定时器 2 的介绍 63
5.2.2 用定时器 2 做波特率发生器（一个完整的程序，在下载板上可以直接测试） 69
5.2.3 定时器 2 的时钟输出功能，在 P1.0 口输出高速脉冲 72
第 6 章　STC89C51RC/RD+系列单片机的掉电模式 74
6.1 PCON 寄存器的高级应用，上电复位标志，进入掉电模式 74
6.2 进入掉电模式后由外部中断唤醒示例程序 75

第 7 章　STC89C51RC/RD+ 系列单片机电气特性 77
第 8 章　STC89C51RC/RD+ 系列单片机开发 / 编程工具说明 79
8.1 编程（ISP）原理使用说明 79
8.2 在系统可编程（ISP）的使用 80
8.3 ISP 软件界面使用说明 81
8.4 用户板如果没有 RS-232,如何用 STC-ISP Ver3.0 PCB 板做 RS-232 通信转换 82
第 9 章　附录 83
9.1 附录 A　STC89LE516AD、58AD、54AD、52AD 系列单片机指南 83
9.1.1 选型 83
9.1.2 特殊功能寄存器映像 84
9.1.3 管脚图 85
9.1.4 典型应用电路 86
9.1.5 扩展 RAM 的使用 87
9.1.6 一个完整的 A/D 转换测试程序 89
9.2 附录 B 为什么少数用户的普通 8051 程序烧录后，不能运行 91
9.3 附录 C STC89C51RC/RD+系列单片机 ISP（DIY）....................................... 92
9.3.1 用户程序区空间和 ISP 程序区空间地址介绍 92
9.3.2 用 STC 专用烧录器烧录用户自己的 ISP 代码进单片机 93
9.3.3 用户写自己的 ISP 程序的格式 95
9.4 附录 D 如何实现运行中不停电自定义下载，无仿真器时方便调试 96
9.5 附录 E Keil C51 高级语言编程的软件如何减少代码长度 97
9.6 附录 F STC89C51RC/RD+系列单片机做仿真器须知 98
9.7 附录 G 典型 MCU-/DSP/uC 复位、电源监控、外部看门狗专用电路 99
9.8 附录 H STC 高性能 SRAM 选型一览表 100
9.9 附录 I STC 单片机配套工具及价格 101
9.10 附录 J STC12C5410AD 系列 1T 8051--- 高速，强抗干扰，4 路 PWM，8 路 10 位 A/D 转换 .. 102
9.11 附录 K 指令系统与程序设计 224
9.12 附录 L 利用 STC 单片机普通 I/O 口驱动 LCD 显示 259
9.13 附录 M STC89 系列单片机和 STC12 系列单片机双 CPU 通信 266
9.14 附录 N 提供过 4000V 快速脉冲干扰辅导服务 272
9.15 附录 O 资料升级历史备忘录 273

STC89C51RC/RD+系列 单片机简介

STC89C51RC/RD+系列单片机是宏晶科技推出的新一代超强抗干扰/高速/低功耗的单片机，指令代码完全兼容传统8051单片机，12时钟/机器周期和6时钟/机器周期可任意选择，最新的D版本内部集成MAX810专用复位电路。

特 点 ：

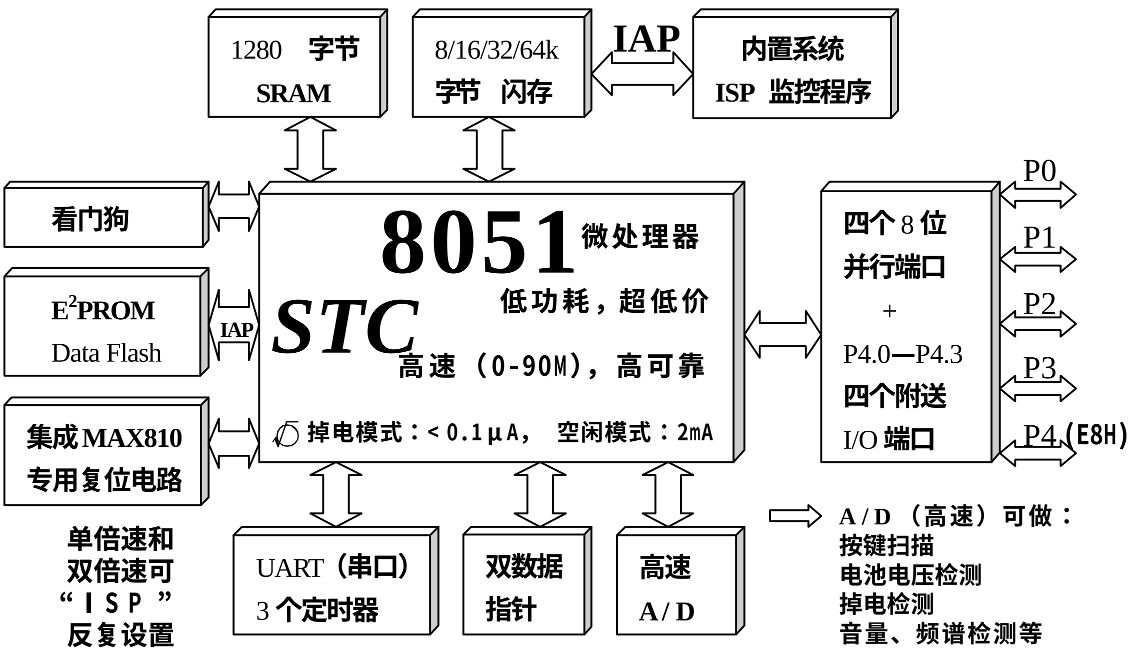
1. 增强型6时钟/机器周期，12时钟/机器周期 8051 CPU
2. 工作电压：5.5V - 3.4V（5V单片机） / 3.8V - 2.0V（3V单片机）
3. 工作频率范围：0 - 40 MHz，相当于普通8051的 0～80MHz.实际工作频率可达48MHz.
4. 用户应用程序空间 4K / 8K / 13K / 16K / 20K / 32K / 64K 字节
5. 片上集成 1280 字节 / 512字节 RAM
6. 通用I/O口（32/36个），复位后为： P1/P2/P3/P4是准双向口/弱上拉（普通8051传统I/O口）
   P0口是开漏输出，作为总线扩展用时，不用加上拉电阻，作为I/O口用时，需加上拉电阻。
7. ISP（在系统可编程）/IAP（在应用可编程），无需专用编程器/仿真器
   可通过串口（P3.0/P3.1）直接下载用户程序，8K程序3秒即可完成一片
8. EEPROM 功能
9. 看门狗
10. 内部集成MAX810专用复位电路（D版本才有），外部晶体20M以下时，可省外部复位电路
11. 共3个16位定时器/计数器，其中定时器0还可以当成2个8位定时器使用
12. 外部中断4路,下降沿中断或低电平触发中断,Power Down模式可由外部中断低电平触发中断方式唤醒
13. 通用异步串行口(UART)，还可用定时器软件实现多个UART
14. 工作温度范围： 0 - 75℃ / -40 - +85℃
15. 封装： LQFP-44,PDIP-40，PLCC-44,PQFP-44,如选择STC89系列,请优先选择LQFP-44封装.

STC89C51RC/RD+系列单片机选型一览表

型号	最高时钟频率Hz		Flash程序存储器字节	RAM数据存储器字节	降低EMI	看门狗	双倍速	P4口	ISP	IAP	EEPROM字节	数据指针	串口UART	中断源	优先级	定时器	A/D
	5V	3V															
STC89C51 RC	0-80M		4K	512	√	√	√	√	√	√	2K+	2	1ch	8	4	3	
STC89C52 RC	0-80M		8K	512	√	√	√	√	√	√	2K+	2	1ch	8	4	3	
STC89C53 RC	0-80M		13K	512	√	√	√	√	√	√		2	1ch	8	4	3	
STC89C54 RD+	0-80M		16K	1280	√	√	√	√	√	√	16K+	2	1ch	8	4	3	
STC89C55 RD+	0-80M		20K	1280	√	√	√	√	√	√	16K+	2	1ch	8	4	3	
STC89C58 RD+	0-80M		32K	1280	√	√	√	√	√	√	16K+	2	1ch	8	4	3	
STC89C516 RD+	0-80M		63K	1280	√	√	√	√	√	√		2	1ch	8	4	3	
STC89LE51 RC		0-80M	4K	512	√	√	√	√	√	√	2K+	2	1ch	8	4	3	
STC89LE52 RC		0-80M	8K	512	√	√	√	√	√	√	2K+	2	1ch	8	4	3	
STC89LE53 RC		0-80M	13K	512	√	√	√	√	√	√		2	1ch	8	4	3	
STC89LE54 RD+		0-80M	16K	1280	√	√	√	√	√	√	16K+	2	1ch	8	4	3	
STC89LE58 RD+		0-80M	32K	1280	√	√	√	√	√	√	16K+	2	1ch	8	4	3	
STC89LE516RD+		0-80M	63K	1280	√	√	√	√	√	√		2	1ch	8	4	3	
STC89LE516AD		0-90M	64K	512	√			√	√			2	1ch	6	4	3	√
STC89LE516X2		0-90M	64K	512	√		√	√	√			2	1ch	6	4	3	√

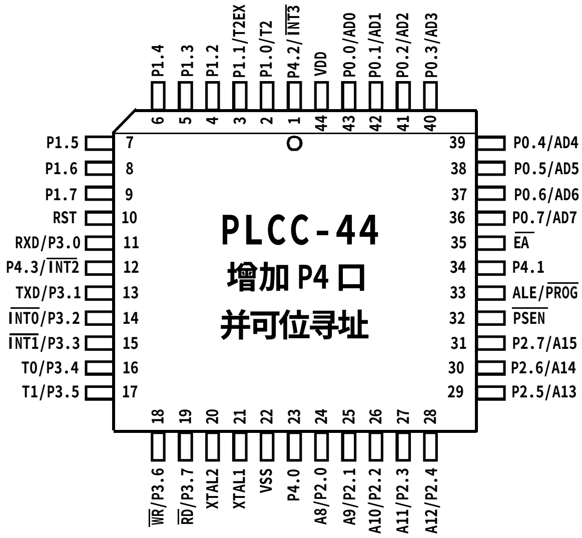
选用STC单片机的理由：降低成本，提升性能，原有程序直接使用,硬件无需改动。STC公司鼓励您放心大胆选用LQFP44小型封装单片机,使您的产品更小，更轻,功耗更低。

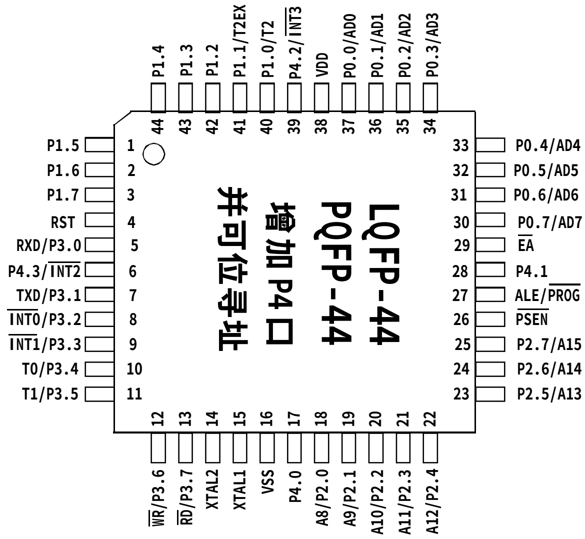
用STC提供的STC-ISP.exe工具将您的2进制代码或16进制代码下载进STC相关的单片机即可。

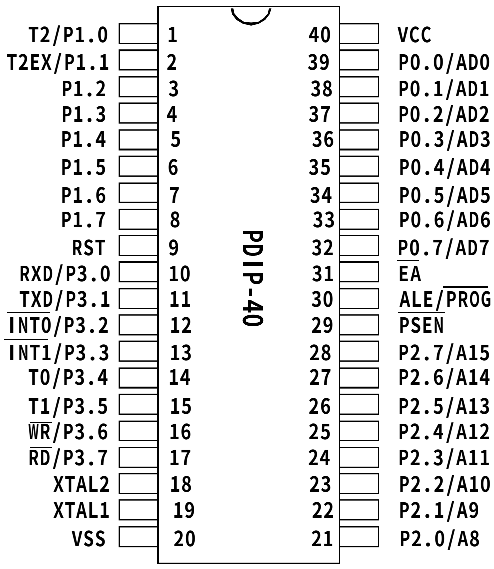


RC/RD+系列为真正的看门狗，缺省为关闭（冷启动），启动后无法关闭，可放心省去外部看门狗。内部Flash擦写次数为100,000次以上，STC89C51RC/RD+系列单片机加密性强,出厂时就已加密。

STC89C51RC / RD+ 系列单片机 管脚图







关于编译器 / 汇编器：

1.任何老的编译器 / 汇编器均可使用
Keil C51 中：Device 选择标准的 Intel8052
头文件包含标准的 <reg52.h>
2.新增特殊功能寄存器如要用到，则用
“sfr”及“sbit”声明地址即可
3.汇编中用“data”, 或“EQU”声明地址

关于仿真及仿真器：

1.任何老的仿真器均可使用
2.老的仿真器仿真他可仿真的基本功能
3.新增特殊功能用 ISP 直接下载程序看结果即可
5.其实现在大部分 STC 用户不用仿真器,用 ISP 就
可调通 64K 程序

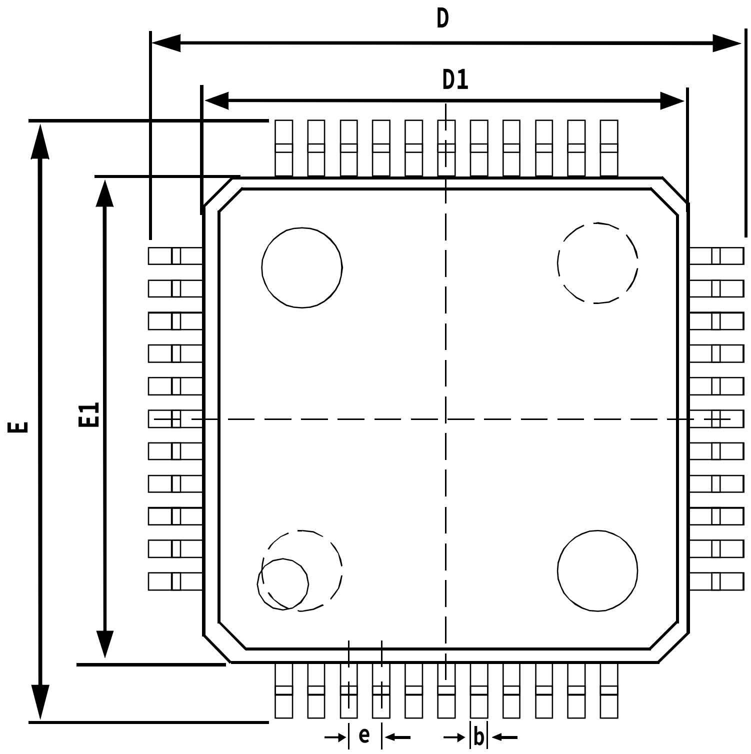
关于工作电压 / 时钟频率:RC/RD+ 系列是真正的 6T 单片机，兼容普通的 12 时钟 / 机器周期

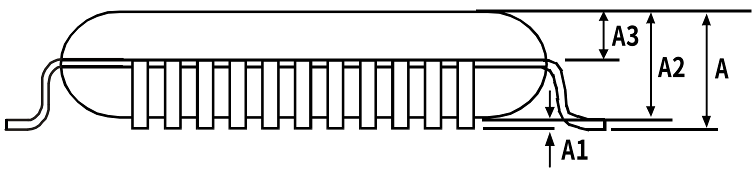
内核实际6T	现有D版本5V单片机,单倍速工作是将外部时钟频率除以2, 降频工作					
工作电压	外部时钟	单倍速相当于普通8052	实际内核运行时钟	双倍速相当于普通8052	实际内核运行时钟	IAP/ISP可以
5.5V - 4.5V	0-40MHz	0-40MHz	0-20MHz	0-80MHz	0-40MHz	读，编程，擦除
5.5V - 3.8V	0-33MHz	0-33MHz	0-16.5M	0-66MHz	0-33MHz	读，编程，擦除
5.5V - 3.6V	0-24MHz	0-24MHz	0-12MHz	0-48MHz	0-24MHz	读，编程，擦除
5.5V - 3.4V	0-20MHz	0-20MHz	0-10MHz	0-40MHz	0-20MHz	读（不要编程/擦除）

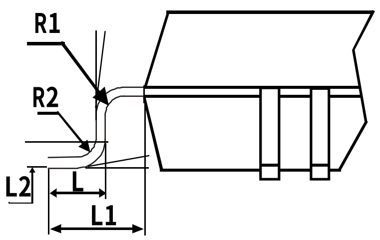
3V: 3.6 - 2.4V(可外部24MHz,双倍速48MHz),2.3-1.9V 时不要进行 IAP 擦除 / 编程

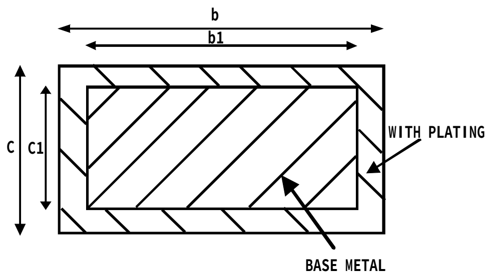
STC 8051 封装尺寸图

LQFP-44 OUTLINE PACKAGE



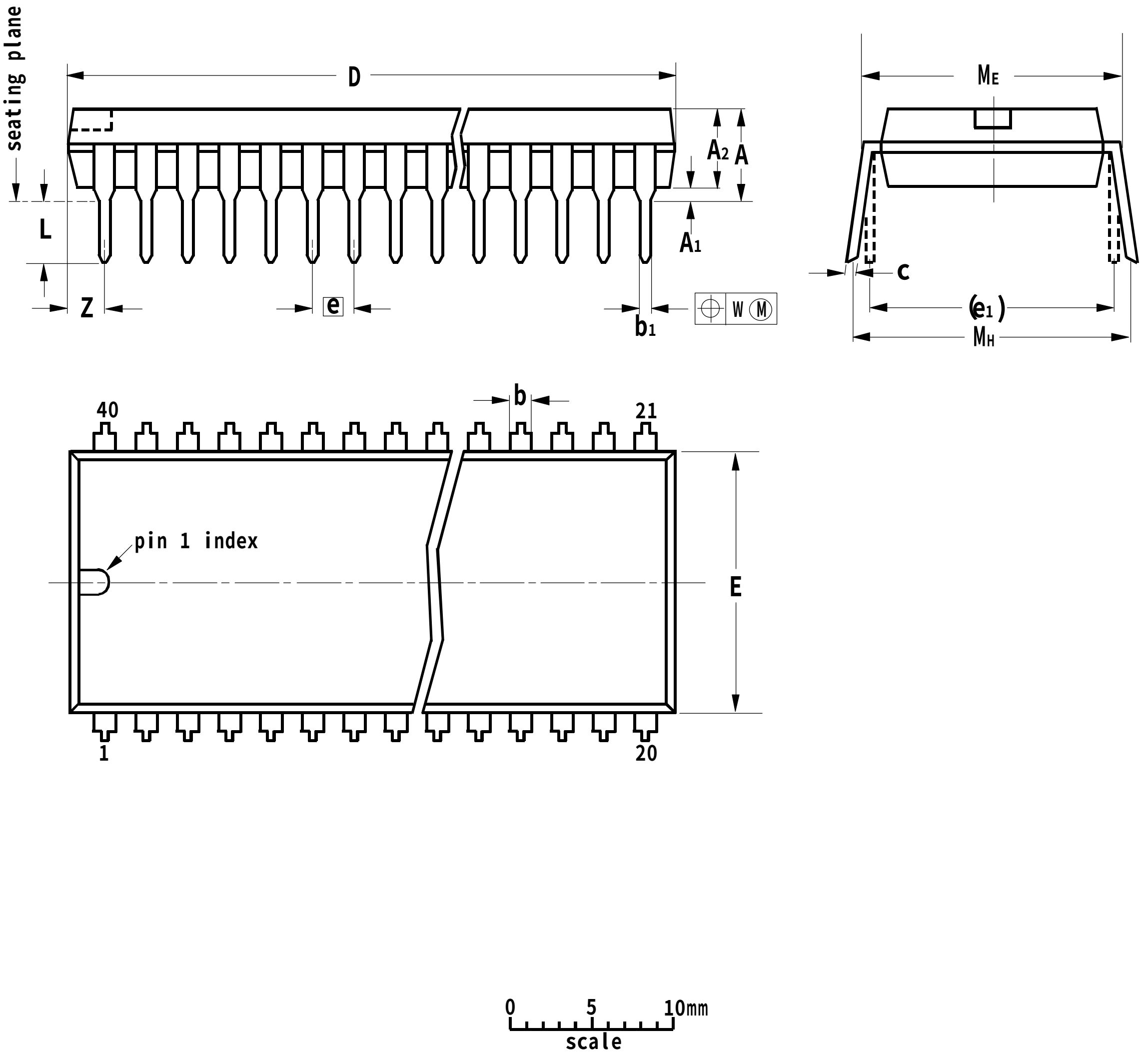






COMMON DIMENSIONS			
(UNITS OF MEASURE = MILLIMETER)			
SYMBOL	MIN	NOM	MAX
A	-	-	1.20
A1	0.05	-	0.15
A2	1.35	1.40	1.45
A3	0.39	0.44	0.49
b	0.31	-	0.44
b1	0.30	0.35	0.40
c	0.13	-	0.18
c1	0.12	0.127	0.134
D	11.80	12.00	12.20
D1	9.90	10.00	10.10
E	11.80	12.00	12.20
E1	9.90	10.00	10.10
e	0.80BSC		
L	0.45	0.60	0.75
L1	1.00REF		
L2	0.25BSC		
R1	0.08	-	-
R2	0.08	-	0.20
S	0.20	-	-
Φ	0°	3.5°	7°
Φ1	0°	–	–
Φ2	11°	12°	13°
Φ3	11°	12°	13°

DIP40: plastic dual in-line package;40 leads(600 mil)



DIMENSIONS(inch dimensions are derived from the original mm dimensions)

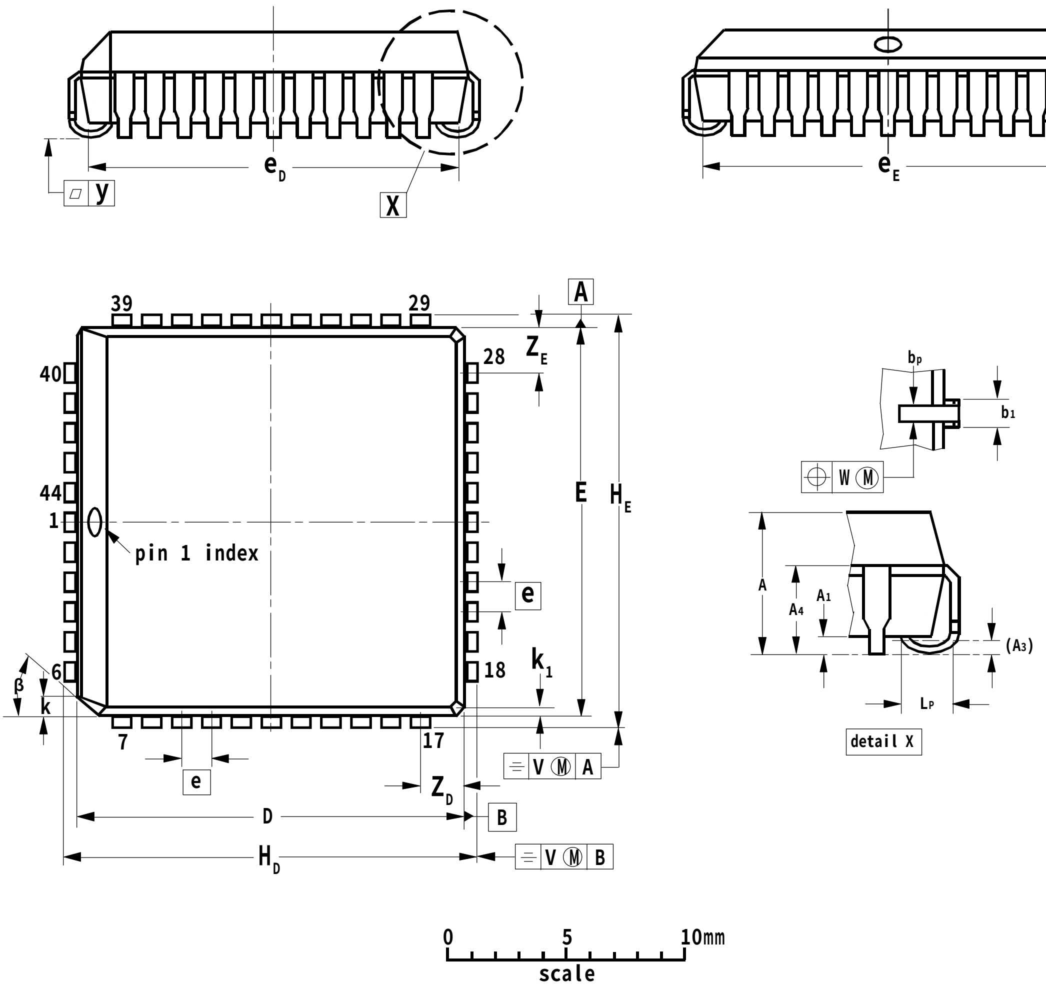
UNIT	A max.	A_1 min.	A_2 max.	b	b_1	c	D[1]	E[1]	e	e_1	L	M_E	M_H	W	Z[1] max.
mm	4.7	0.51	4.0	1.70 1.14	0.53 0.38	0.36 0.23	52.5 51.5	14.1 13.7	2.54	15.24	3.60 3.05	15.8 15.24	17.42 15.90	0.254	2.25
inches	0.19	0.020	0.16	0.067 0.045	0.021 0.015	0.014 0.009	2.067 2.028	0.56 0.54	0.10	0.60	0.14 0.12	0.62 0.60	0.69 0.63	0.01	0.089

Note

1.Plastic or metal protrusion of 0.25 mm maximum per side are not included

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT129-1	051G08	MO-015	SC-511-40			~~95-01-14~~ 99-12-27

PLCC44: plastic leaded chip carrier;44 leads



DIMENSIONS(millimetre dimensions are derived from the original inch dimensions)

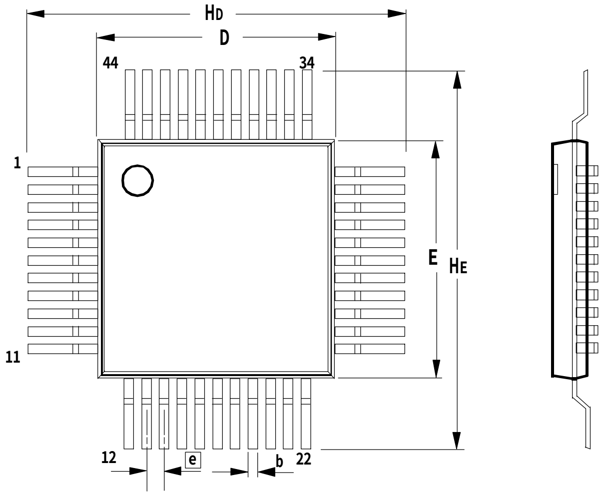
UNIT	A	A_1 max.	A_3	A_4 max.	b_p	b_1	D(1)	E(1)	e	e_D	e_E	H_D	H_E	k	k_1 max.	L_p	v	w	y	Z_D(1) max.	Z_E(1) max.	β
mm	4.57 4.19	0.51	0.25	3.05	0.53 0.33	0.81 0.66	16.66 16.51	16.66 16.51	1.27	16.00 14.99	16.00 14.99	17.65 17.40	17.65 17.40	1.22 1.07	0.51	1.44 1.02	0.18	0.18	0.10	2.16	2.16	45°
inches	0.180 0.165	0.020	0.01	0.12	0.021 0.013	0.032 0.026	0.656 0.650	0.656 0.650	0.05	0.630 0.590	0.630 0.590	0.695 0.685	0.695 0.685	0.048 0.042	0.020	0.057 0.040	0.007	0.007	0.004	0.085	0.085	

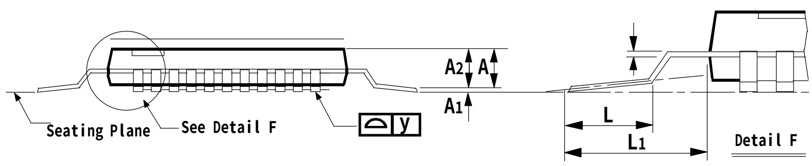
Note

1.Plastic or metal protrusions of 0.01 inches maximum per side are not included

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT187-2	112E10	MO-047				~~97-12-16~~ 99-12-27

PQFP44





Symbol	Dimension in inch			Dimension in mm		
	Min.	Nom.	Max.	Min.	Nom.	Max.
A	--	--	--	--	--	--
A_1	0.002	0.01	0.02	0.05	0.25	0.5
A_2	0.075	0.081	0.087	1.90	2.05	2.20
b	0.01	0.014	0.018	0.25	0.35	0.45
c	0.004	0.006	0.010	0.101	0.152	0.254
D	0.390	0.394	0.398	9.9	10.00	10.1
E	0.390	0.394	0.398	9.9	10.00	10.1
e	0.025	0.031	0.036	0.635	0.80	0.952
H_D	0.510	0.520	0.530	12.95	13.2	13.45
H_E	0.510	0.520	0.530	12.95	13.2	13.45
L	0.025	0.031	0.037	0.65	0.8	0.95
L_1	0.051	0.063	0.075	1.295	1.6	1.905
y	--	--	0.003	--	--	0.08
θ	0°	--	7°	0°	--	7°

Notes:

1.Dimension D & E do not include interlead flash.

2.Dimension b does not include dambar protrusion/intrusion.

3.Controlling dimension Millimeter

4.General appearance spec. should be based on final visual inspection spec.

STC89C51RC/RD+系列 单片机 命名规则

STC89xxxxxx—40x-xxxx

何种封装：如
PDIP，PLCC，PQFP

工作温度范围：
I：工业级，-40℃ ～ +85℃
C：商业级，0℃ ～ 70℃

工作频率:
25：工作频率可到25MHz
40：工作频率可到40MHz
50：工作频率可到50MHz

RAM大小:
RC: RAM为512
RD+: RAM为1280

程序空间大小，如：
51是4K字节， 52是8K字节，53是15K字节
54是16K字节，58是32K字节,516是64K字节

工作电压:
C： 5.5V～3.8V
LE：2.4V～3.8V

STC 12T/6T 8051

超低功耗 ---- STC89C51RC / RD+ 系列单片机

1．掉电模式：

典型功耗　< 0.1uA，　可由外部中断唤醒，中断返回后，继续执行原程序

2．空闲模式：

典型功耗　2mA

3．正常工作模式：

典型功耗　4mA - 7mA

4．掉电模式可由外部中断唤醒，适用于水表、气表等电池供电系统及便携设备

降低单片机对外部的电磁辐射（EMI)--- 三大措施

1．禁止 ALE 时钟信号输出:

RC/RD+ 系列 8051 单片机 扩展 RAM 管理及禁止 ALE 输出 特殊功能寄存器　只写

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
AUXR	8Eh	Auxiliary Register 0	-	-	-	-	-	-	EXTRAM	ALEOFF	xxxx,xx00

禁止 ALE 信号输出(应用示例供参考,汇编语言):

MOV　AUXR，　#00000001B;　ALEOFF 位置“1”,禁止 ALE 时钟输出

2．外部时钟频率降一半，6T 模式：传统的 8051 为每个机器周期 12 时钟，如将 STC 的增强型 8051 单片机在 ISP 烧录程序时设为双倍速（即 6T 模式，每个机器周期 6 时钟），则可将单片机外部时钟频率降低一半，有效的降低单片机时钟对外界的辐射

3．单片机内部时钟振荡器增益降低一半：　在 ISP 烧录程序时将 OSCDN 设为 1/2 gain 可以有效的降低单片机时钟高频部分对外界的辐射,单片机外部晶振频率<16MHz 时，可将 OSCDN 设为 1/2 gain,有利于降低 EMI，16M 以上选择 full gain。

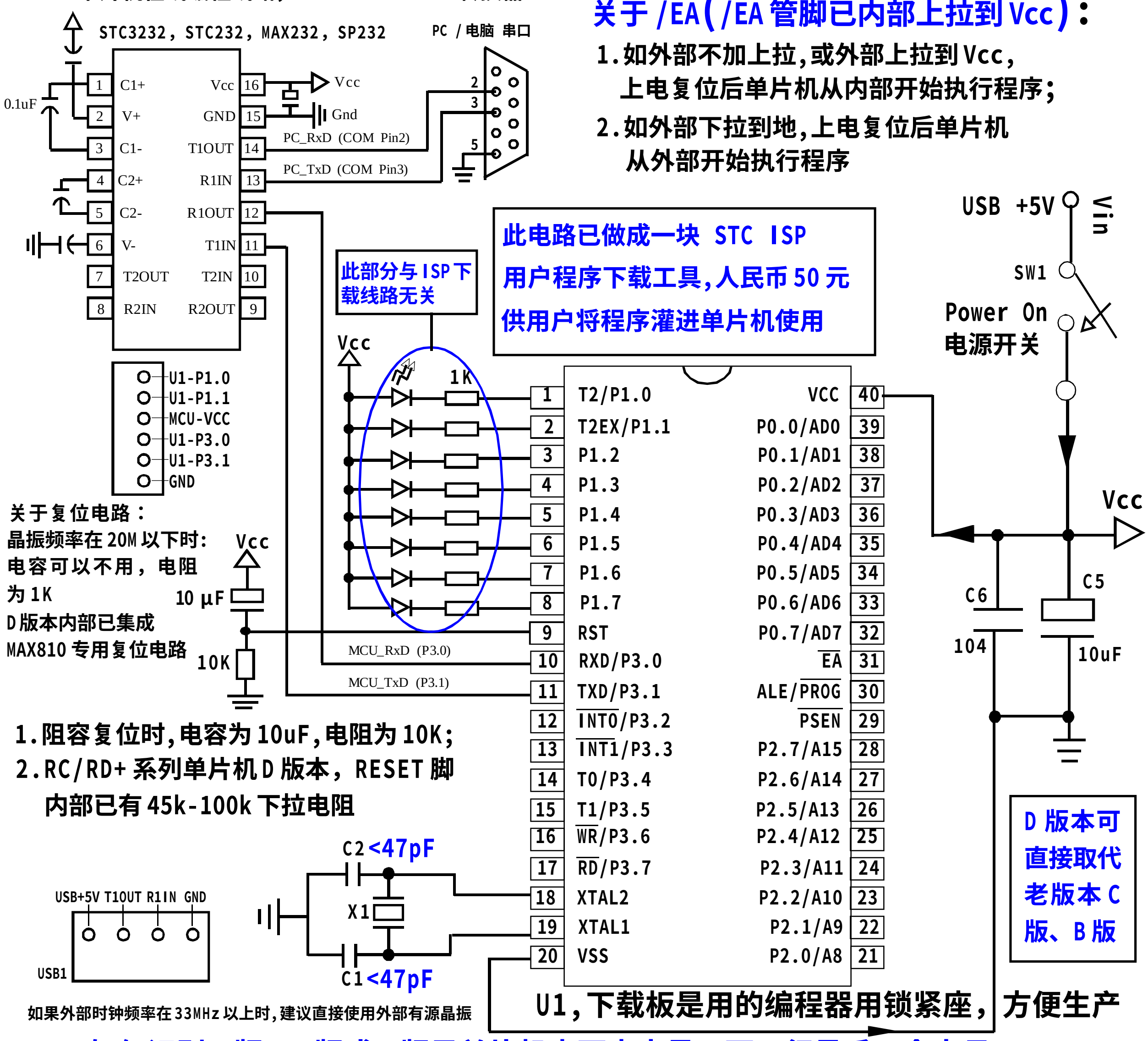
STC 单片机典型应用电路(89C51RC/RD+ 系列,D 版)

STC 单片机在线编程线路， *STC* RS-232 转换器

关于 /EA(/EA 管脚已内部上拉到 Vcc)：

1.如外部不加上拉,或外部上拉到 Vcc,
  上电复位后单片机从外部开始执行程序;

2.如外部下拉到地,上电复位后单片机
  从外部开始执行程序



关于复位电路：
晶振频率在 20M 以下时：电容可以不用，电阻为 1K
D 版本内部已集成 MAX810 专用复位电路

1.阻容复位时,电容为 10uF,电阻为 10K;

2.RC/RD+ 系列单片机 D 版本，RESET 脚
  内部已有 45k-100k 下拉电阻

如果外部时钟频率在 33MHz 以上时,建议直接使用外部有源晶振

U1,下载板是用的编程器用锁紧座，方便生产

※ 如何识别 D 版、C 版或 B 版见单片机表面文字最下面一行最后一个字母

关于晶振电路:

OSCDN,晶体振荡器增益控制 = full gain						
X1	2-25MHz	26-30MHz	31-35MHz	36-39MHz	40-43MHz	44-48MHz
C1,C2	<= 47pF	<=10pF	<=10pF	<=10pF	<=10pF	<=5pF
R1	不用	6.8K	5.1K	4.7K	3.3K	3.3K

OSCDN(OSC Control),振荡器增益 = 1/2 gain						
X1	2-25MHz	26-30MHz	31-35MHz	36-39MHz	40-43MHz	44-48MHz
C1,C2	<=47pF	<=5pF	不用	不用	不用	不用
R1	不用	6.8K	5.1K	4.7K	3.3K	3.3K

STC89系列D版本的单片机正常工作时的时钟频率

推荐工作时钟频率(总线) STC单片机 RC/RD+系列 (I/O方式可到40M/80M)	内部振荡器产生时钟，外接晶体		外部时钟直接输入，由XTAL1输入	
	12时钟模式	6时钟模式	12时钟模式	6时钟模式
5.0V 单片机	2MHz - 48MHz	2MHz - 36MHz	2MHz - 48MHz	2MHz - 36MHz
3.3V 单片机	2MHz - 48MHz	2MHz - 32MHz	2MHz - 36MHz	2MHz - 18MHz

特殊功能寄存器映像　SFR Mapping

STC89C51RC, STC89C52RC, STC89C53RC, STC89C54RD+,STC89C58RD+,STC89C516RD+
STC89LE51RC,STC89LE52RC,STC89LE53RC,STC89LE54RD+,STC89LE58RD+,STC89LE516RD

	Bit Addressable	Non Bit Addressable							
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	
F8h									FFh
F0h	B 0000,0000								F7h
E8h	P4 xxxx,1111								EFh
E0h	ACC 0000,0000	WDT_CONTR xx00,0000	ISP_DATA 1111,1111	ISP_ADDRH 0000,0000	ISP_ADDRL 0000,0000	ISP_CMD 1111,1000	ISP_TRIG xxxx,xxxx	ISP_CONTR 000x,x000	E7h
D8h									DFh
D0h	PSW 0000,0000								D7h
C8h	T2CON 0000,0000	T2MOD xxxx,xx00	RCAP2L 0000,0000	RCAP2H 0000,0000	TL2 0000,0000	TH2 0000,0000			CFh
C0h	XICON 0000,0000								C7h
B8h	IP xx00,0000	SADEN 0000,0000							BFh
B0h	P3 1111,1111							IPH 0000,0000	B7h
A8h	IE 0000,0000	SADDR 0000,0000							AFh
A0h	P2 1111,1111		AUXR1 xxxx,0xx0						A7h
98h	SCON 0000,0000	SBUF xxxx,xxxx							9Fh
90h	P1 1111,1111								97h
88h	TCON 0000,0000	TMOD 0000,0000	TL0 0000,0000	TL1 0000,0000	TH0 0000,0000	TH1 0000,0000	AUXR xxxx,xx00		8Fh
80h	P0 1111,1111	SP 0000,0111	DPL 0000,0000	DPH 0000,0000				PCON 00x1,0000	87h
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	

可位寻址　　　不可位寻址

STC89C51RC/RD+系列8051 单片机内核特殊功能寄存器 C51 Core SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
ACC	E0h	Accumulator									0000,0000
B	F0h	B Register									0000,0000
PSW	D0h	Program Status Word	CY	AC	F0	RS1	RS0	OV	F1	P	0000,0000
SP	81h	Stack Pointer									0000,0111
DPL	82h	Data Pointer Low Byte									0000,0000
DPH	83h	Data Pointer High Byte									0000,0000

STC89C51RC/RD+系列8051 单片机系统管理特殊功能寄存器 System Management SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
PCON	87h	Power Control	SMOD	SMOD0	-	POF	GF1	GF0	PD	IDL	00x1,0000
AUXR	8Eh	Auxiliary Register 0	-	-	-	-	-	-	EXTRAM	ALEOFF	xxxx,xx00
AUXR1	A2h	Auxiliary Register 1	-	-	-	-	GF2	-	-	DPS	xxxx,0xx0

不同：STC89LE516AD / 89LE516X2 系列单片机没有 EXTRAM 控制位.

STC89C51RC/RD+系列8051 单片机 中断 特殊功能寄存器 Interrupt SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
IE	A8h	Interrupt Enable	EA	-	ET2	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	-	-	PT2	PS	PT1	PX1	PT0	PX0	xx00,0000
IPH	B7h	Interrupt Priority High	PX3H	PX2H	PT2H	PSH	PT1H	PX1H	PT0H	PX0H	0000,0000
TCON	88h	Timer / Counter 0 and 1 Control	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000,0000
SCON	98h	Serial Control	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000,0000
T2CON	C8h	Timer / Counter 2 Control	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2#	CP/RL2#	0000,0000
XICON	C0h	Auxiliary Interrupt Control	PX3	EX3	IE3	IT3	PX2	EX2	IE2	IT2	0000,0000

不同：STC89LE516AD 系列单片机没有(XICON,PX3H,PX2H),因为 P4.2/P4.3 无中断.

STC89C51RC/RD+系列8051 单片机 I/O 口 特殊功能寄存器 Port SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P0	80h	8-bit Port 0	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	1111,1111
P1	90h	8-bit Port 1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	1111,1111
P2	A0h	8-bit Port 2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	1111,1111
P3	B0h	8-bit Port 3	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	1111,1111
P4	E8h	4-bit Port 4	-	-	-	-	P4.3	P4.2	P4.1	P4.0	xxxx,1111

不同：STC89LE516AD / 89LE516X2 系列单片机 P4 口地址为 C0h,而不是 E8h.

STC89C51RC/RD+系列 8051 单片机 串行口 特殊功能寄存器 Serial I/O Port SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
SCON	98h	Serial Control	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000,0000
SBUF	99h	Serial Data Buffer									xxxx,xxxx
SADEN	B9h	Slave Address Mask									0000,0000
SADDR	A9h	Slave Address									0000,0000

STC89C51RC/RD+系列 8051 单片机 定时器 特殊功能寄存器 Timer SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
TCON	88h	Timer / Counter 0 and 1 Control	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000,0000
TMOD	89h	Timer / Counter 0 and 1 Modes	GATE GATE1	C/T# C/T1#	M1 M1_1	M0 M1_0	GATE GATE0	C/T# C/T0#	M1 M0_1	M0 M0_0	0000,0000
TL0	8Ah	Timer / Counter 0 Low Byte									0000,0000
TH0	8Ch	Timer / Counter 0 High Byte									0000,0000
TL1	8Bh	Timer / Counter 1 Low Byte									0000,0000
TH1	8Dh	Timer / Counter 1 High Byte									0000,0000
T2CON	C8h	Timer / Counter 2 Control	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2#	CP/RL2#	0000,0000
T2MOD	C9h	Timer / Counter 2 Mode	-	-	-	-	-	-	T2OE	DCEN	xxxx,xx00
RCAP2L	CAh	Timer / Counter 2 Reload/Capture Low Byte									0000,0000
RCAP2H	CBh	Timer / Counter 2 Reload/Capture High Byte									0000,0000
TL2	CCh	Timer / Counter 2 Low Byte									0000,0000
TH2	CDh	Timer / Counter 2 High Byte									0000,0000

STC89C51RC/RD+系列 8051 单片机 看门狗定时器 特殊功能寄存器 Watch Dog Timer SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
WDT_CONTR	E1h	Watch-Dog-Timer Control register	-	-	EN_WDT	CLR_WDT	IDLE_WDT	PS2	PS1	PS0	xx00,0000

STC89C51RC/RD+系列 8051 单片机 ISP/IAP 特殊功能寄存器 ISP/IAP SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
ISP_DATA	E2h	ISP/IAP Flash Data Register									1111,1111
ISP_ADDRH	E3h	ISP/IAP Flash Address High									0000,0000
ISP_ADDRL	E4h	ISP/IAP Flash Address Low									0000,0000
ISP_CMD	E5h	ISP/IAP Flash Command Register	-	-	-	-	-	MS2	MS1	MS0	xxxx,x000
ISP_TRIG	E6h	ISP/IAP Flash Command Trigger									xxxx,xxxx
ISP_CONTR	E7h	ISP/IAP Control Register	ISPEN	SWBS	SWRST	-	-	WT2	WT1	WT0	000x,x000

中断

RC/RD+系列8051 单片机 中断 特殊功能寄存器 Interrupt SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
IE	A8h	Interrupt Enable	EA	-	ET2	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	-	-	PT2	PS	PT1	PX1	PT0	PX0	xx00,0000
IPH	B7h	Interrupt Priority High	PX3H	PX2H	PT2H	PSH	PT1H	PX1H	PT0H	PX0H	0000,0000
TCON	88h	Timer / Counter 0 and 1 Control	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000,0000
SCON	98h	Serial Control	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000,0000
T2CON	C8h	Timer / Counter 2 Control	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2#	CP/RL2#	0000,0000
XICON	C0h	Auxiliary Interrupt Control	PX3	EX3	IE3	IT3	PX2	EX2	IE2	IT2	0000,0000

中断与普通8052完全兼容，优先级可设为4级，另增加2个外部中断INT2/P4.3,INT3/P4.2。

Interrupt Source 中断源	Vector Address 中断 向量地址	Polling Sequence 中断 查询次序	中断 优先级设置	优先级0 最低	优先级1	优先级2	优先级3 最高	Interrupt Request 中断请求
/INT0	0003H	0(最优先)	PX0H,PX0	0,0	0,1	1,0	1,1	IE0
Timer 0	000BH	1	PT0H,PT0	0,0	0,1	1,0	1,1	TF0
/INT1	0013H	2	PX1H,PX1	0,0	0,1	1,0	1,1	IE1
Timer 1	001BH	3	PT1H,PT1	0,0	0,1	1,0	1,1	TF1
UART	0023H	4	PSH, PS	0,0	0,1	1,0	1,1	RI + TI
Timer 2	002BH	5	PT2H,PT2	0,0	0,1	1,0	1,1	TF2 + EXF2
/INT2	0033H	6	PX2H,PX2	0,0	0,1	1,0	1,1	IE2
/INT3	003BH	7(最低)	PX3H,PX3	0,0	0,1	1,0	1,1	IE3

XICON（扩展中断控制）寄存器，控制外部中断INT2/INT3

Name	Function
PX3	置位表明外部中断3的优先级为高,优先级最终由[PXH3，PX3]=[0，0]；[0，1]；[1，0]；[1，1]来决定
EX3	如被设置成1，允许外部中断3中断；如被清成0，禁止外部中断3中断。
IE3	外部中断3中断请求标志位，中断条件成立后，IE3=1,可由硬件自动清零。
IT3	当此位由软件置位时,外部中断3为下降沿触发中断；当此位由软件清零时,为低电平触发中断。
PX2	置位表明外部中断2的优先级为高，优先级最终由[PXH2，PX2]=[0，0]；[0，1]；[1，0]；[1，1]来决定
EX2	如被设置成1，允许外部中断2中断；如被清成0，禁止外部中断2中断。
IE2	外部中断2中断请求标志位，中断条件成立后，IE2=1,可由硬件自动清零。
IT2	当此位由软件置位时,外部中断2为下降沿触发中断；当此位由软件清零时,为低电平触发中断。
PXH3	外部中断3最高中断优先级设置位置高，优先级最终由[PXH3，PX3]=[0，0]；[0，1]；[1，0]；[1，1]来决定
PXH2	外部中断2最高中断优先级设置位置高，优先级最终由[PXH2，PX2]=[0，0]；[0，1]；[1，0]；[1，1]来决定

降低单片机对系统的电磁干扰（EMI)--- 三大措施

1. 禁止 ALE 信号输出，适用型号：
STC89C51RC,STC89C52RC,STC89C53RC,STC89LE51RC,STC89LE52RC,STC89LE53RC
STC89C54RD+,STC89C58RD+,STC89C516RD+,STC89LE54RD+,STC89LE58RD+,STC89LE516RD+
STC89LE516AD / X2 系列 （注：此系列单片机无 EXTRAM 控制位）

RC/RD+系列8051 单片机 扩展RAM管理及禁止ALE输出 特殊功能寄存器 只写

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
AUXR	8Eh	Auxiliary Register 0	-	-	-	-	-	-	EXTRAM	ALEOFF	xxxx,xx00

禁止 ALE 信号输出(应用示例供参考,C 语言)：

```
sfr       AUXR  =    0x8e;      /* 声明AUXR 寄存器的地址 */
AUXR  =   0x01;
 /* ALEOFF 位置 1,禁止 ALE 信号输出,提升系统的 EMI 性能,复位后为 0,ALE 信号正常输出 */
```

禁止 ALE 信号输出(应用示例供参考,汇编语言)：

```
AUXR  EQU   8Eh   ;         或       AUXR  DATA  8Eh
MOV   AUXR,#00000001B;   ALEOFF 位置“1”,禁止 ALE 信号输出,提升了系统的 EMI 性能
```

2. 外部时钟频率降一半，6T 模式： 传统的 8051 为每个机器周期 12 时钟，如将 STC 的增强型 8051 单片机在 ISP 烧录程序时设为双倍速（及 6T 模式，每个机器周期 6 时钟），则可将单片机外部时钟频率降低一半，有效的降低单片机时钟对外界的干扰

3. 单片机内部时钟振荡器增益降低一半： 在 ISP 烧录程序时将 OSCDN 设为 1/2 gain 可以有效的降低单片机时钟高频部分对外界的辐射,但此时外部晶振频率尽量不要高于 16MHz。

STC89C51RC/RD+ 系列单片机扩展 RAM 的使用
STC89C51RC/RD+ 系列单片机扩展 RAM 的禁止

适用型号:
STC89C51RC,STC89C52RC,STC89C53RC,STC89LE51RC,STC89LE52RC,STC89LE53RC
STC89C54RD+,STC89C58RD+,STC89C516RD+,STC89LE54RD+,STC89LE58RD+,STC89LE516RD+

普通 89C51，89C52 系列单片机的内部 RAM 只有 128（89C51）/256（89C52）供用户使用
1).低 128 字节的内部 RAM（地址：00H～7FH），可直接寻址或间接寻址，(data/idata)
2).高 128 字节的内部 RAM（地址：80H～FFH），只能间接寻址（普通 89C51 没有），(idata)
3).特殊功能寄存器 SFR（地址：80H～FFH），只能直接寻址，(data)

特殊功能寄存器 SFR 和高 128 字节的内部 RAM 是通过寻址方式来区分的，传统的 8051 系列单片机只有 128-256 字节 RAM 供用户使用，在此情况下 STC 公司响应广大用户的呼声，在一些单片机内部增加了扩展 RAM。STC89C58RD+ 系列单片机扩展了 1024 个字节 RAM，共 1280 字节 RAM；STC89C52RC 系列扩展了 256 个字节 RAM，共 512 字节 RAM。访问内部扩展 RAM 时，不影响 P0 口 /P2 口 /P3.6/P3.7/ALE。

RC/RD+ 系列 8051 单片机 扩展 RAM 管理及禁止 ALE 输出 特殊功能寄存器 只写

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
AUXR	8Eh	Auxiliary Register 0	-	-	-	-	-	-	EXTRAM	ALEOFF	xxxx,xx00

Symbol 符号 Function 功能

EXTRAM Internal/External RAM access 内部 / 外部 RAM 存取

0: 内部扩展的 EXT_RAM 可以存取.

RD+ 系列单片机

在 00H 到 3FFH 单元(1024 字节),使用 MOVX @DPTR 指令访问,超过 400H 的地址空间总是访问外部数据存储器（含 400H 单元），MOVX @Ri 只能访问 00H 到 FFH 单元

RC 系列单片机

在 00H 到 FFH 单元(256 字节),使用 MOVX @DPTR 指令访问,超过 100H 的地址空间总是访问外部数据存储器（含 100H 单元），MOVX @Ri 只能访问 00H 到 FFH 单元

1: External data memory access.

外部数据存储器存取,禁止访问内部扩展 RAM，此时 MOVX @DPTR / MOVX @Ri 的使用同普通 8052 单片机

ALEOFF Disable/enable ALE.

0: ALE is emitted at a constant rate of 1/3 the oscillator frequency in 6 clock mode, 1/6 fosc in 12 clock mode

ALE 脚输出固定的 1/6 晶振频率信号在 12 时钟模式时,在 6 时钟模式时输出固定的 1/3 晶振频率信号.

1: ALE is active only during a MOVX or MOVC instruction.

ALE 脚仅在执行 MOVX or MOVC 指令时才输出信号,好处是：降低了系统对外界的 EMI.

注解：STC89LE516AD,STC89LE516X2 系列无 EXTRAM 控制位,仅有 ALEOFF 控制位。
STC89LE516AD/X2 系列用 MOVX A, @Ri / MOVX @Ri, A 指令固定访问内部扩展的 EXTRAM,
用 MOVX A, @DPTR / MOVX @DPTR, A 指令固定访问外部 RAM.

应用示例供参考（汇编）：

访问内部扩展的 EXTRAM

```
;新增特殊功能寄存器声明(汇编方式)
AUXR    DATA    8EH;          或者用     AUXR  EQU    8EH     定义
MOV     AUXR,   #00000000B;    EXTRAM 位清为"0",  其实上电复位时此位就为"0".
    ;MOVX   A,   @DPTR  /  MOVX  @DPTR, A 指令可访问内部扩展的 EXTRAM
        ;RD+ 系列为(00H  -  3FFH,共 1024 字节)
        ;RC 系列为(00H  -  FFH,共 256 字节)
    ;MOVX   A,   @Ri  /  MOVX   A, @Ri  指令可直接访问内部扩展的 EXTRAM
        ;使用此指令 RD+ 系列  只能访问内部扩展的 EXTRAM(00H  -  FFH,共 256 字节)
;写芯片内部扩展的 EXTRAM
    MOV     DPTR,   #address
    MOV     A,  #value
    MOVX    @DPTR,  A
;读芯片内部扩展的 EXTRAM
    MOV     DPTR,   #address
    MOVX    A,  @DPTR
```

RD+ 系列

; 如果 #address < 400H，则在 EXTRAM 位为"0"时，访问物理上在内部，逻辑上在外部的此 EXTRAM

; 如果 #address >= 400H，则总是访问物理上外部扩展的 RAM 或 I/O 空间（400H--FFFFH）

RC 系列

; 如果 #address < 100H，则在 EXTRAM 位为"0"时，访问物理上在内部，逻辑上在外部的此 EXTRAM

; 如果 #address >= 100H，则总是访问物理上外部扩展的 RAM 或 I/O 空间（100H--FFFFH）

禁止访问内部扩展的 EXTRAM ,以防冲突

```
MOV     AUXR,   #00000010B;   EXTRAM 控制位设置为"1",  禁止访问 EXTRAM,以防冲突
```

有些用户系统因为外部扩展了 I/O 或者用片选去选多个 RAM 区,有时与此内部扩展的 EXTRAM 逻辑地址上有冲突，将此位设置为"1", 禁止访问此内部扩展的 EXTRAM 就可以了.

大实话 ： 其实不用设置 AUXR 寄存器即可直接用 MOVX @DPTR 指令访问此内部扩展的 EXTRAM,超过此 RAM 空间,将访问片外单元.如果系统外扩了 SRAM,而实际使用的空间小于 1024/256 字节,则可直接将此 SRAM 省去,比如省去 STC62WV256,IS62C256,UT6264 等. 另外尽量用 MOVX A, @Ri / MOVX @Ri, A指令访问此内部扩展的 EXTRAM,这样只能访问 256 字节的扩展 EXTRAM,但可与很多单片机兼容。如 STC89LE516AD/X2 系列 MOVX A, @Ri / MOVX @Ri, A 指令只能固定访问内部扩展的 EXTRAM, MOVX A, @DPTR / MOVX @DPTR, A 指令固定访问外部 RAM.

应用示例供参考（C 语言）：

```
/*  访问内部扩展的 EXTRAM  */
/*  RD+ 系列为(00H  -  3FFH, 共 1024 字节扩展的 EXTRAM)  */
/*  RC  系列为(00H  -   FFH, 共 256 字节扩展的 EXTRAM)   */
    /*  新增特殊功能寄存器声明(C 语言方式)    */
    sfr AUXR     =    0x8e     /*  如果不需设置 AUXR 就不用声明 AUXR        */
    AUXR    =   0x00;   /*  0000,0000   EXTRAM 位清 0, 其实上电复位时此位就为 0  */
    unsigned char   xdata   sum, loop_counter, test_array[128];
    /*   将变量声明成 xdata 即可直接访问此内部扩展的 EXTRAM      */
```

```
/*  写芯片内部扩展的 EXTRAM */
        sum =   0;
        loop_counter    =   128;
        test_array[0]   =   5;
/*  读芯片内部扩展的 EXTRAM */
        sum =   test_array[0];
    /*  RD+ 系列:
                如果 #address < 400H，则在 EXTRAM 位为 "0" 时，访问物理上在内部，逻辑
            上在外部的此 EXTRAM
                如果 #address>=400H，则总是访问物理上外部扩展的 RAM 或 I/O 空间（400H-FFFFH）
        RC 系列:
                如果 #address < 100H，则在 EXTRAM 位为 "0" 时，访问物理上在内部，逻辑
            上在外部的此 EXTRAM
                如果 #address>=100H，总是访问物理上外部扩展的 RAM 或 I/O 空间（100H--FFFFH）
    */
```

禁止访问内部扩展的 EXTRAM，以防冲突

AUXR = 0x02; /* 0000,0010, EXTRAM 位设为"1"，禁止访问 EXTRAM,以防冲突 */

有些用户系统因为外部扩展了 I/O 或者用片选去选多个 RAM 区,有时与此内部扩展的 EXTRAM 逻辑上有冲突，将此位设置为"1"，禁止访问此内部扩展的 EXTRAM 就可以了.

AUXR 是只写寄存器

所谓只写，就是直接用"MOV AUXR, #data"去写,而不要用含读的操作如"或，与，入栈"因为他不让你读，如去读，读出的数值不确定，用含读的操作如"或，与，入栈"，会达不到需要的效果。

单片机 C 版本 / D 版本和以前版本的区别（关于内部扩展 RAM）

传统的 8051，内部无扩展 RAM，而 STC89C51RC/RD+ 系列单片机内部均已扩展了 RAM,少数客户的老产品 P0/P2 是作为总线用的而不是作为普通 I/O 口用，有些需要用软件关闭此内部扩展 RAM。而客户的源程序早已遗失，或开发工程师早已离职，所以 STC89C51RC/RD+ 系列单片机为了解决此问题，推出 C 版本以供用户在 ISP 下载程序时就可选择关闭此内部扩展 RAM，以达到完全兼容以前的老产品的目的。

一般不要在 ISP 下载程序时就选择关闭此内部扩展 RAM，因为流行用法是复位后缺省是允许访问扩展 RAM，复位后 AUXR.1/AUXR.EXTRAM = 0，选择关闭此内部扩展 RAM,则本来是：

	在 ISP 下载程序时选择"允许访问内部扩展 RAM"	在 ISP 下载程序时选择"禁止访问内部扩展 RAM"
AUXR.1/AUXR.EXTRAM = 0	是允许访问内部扩展 RAM	是禁止访问内部扩展 RAM
AUXR.1/AUXR.EXTRAM = 1	是禁止访问内部扩展 RAM	是允许访问内部扩展 RAM

另 STC89C51RC/RD+ 系列单片机 C 版本以前的单片机"AUXR 寄存器是只写特性"，现 C 版本及以后的版本将都是既可以读又可以写。

STC89C58RD+ 系列单片机内部扩展 RAM 演示程序

```
;/* ---  STC International Limited ---------------------------------------- */
;/* ---  宏晶科技 姚永平  设计 2006/1/6   V1.0 --------------------------- */
;/* ---  演示 STC89C/LE51RC/RD+ 系列 MCU 内部扩展 RAM 演示程序 -------------- */
;/* ---  Mobile: 13922805190 --------------------------------------------- */
;/* ---  Fax: 0755-82944243 ---------------------------------------------- */
;/* ---  Tel: 0755-82948409 ---------------------------------------------- */
;/* ---  Web: www.mcu-memory.com ----------------------------------------- */
;/* ---  本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过 ---------- */
;/* --- 如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 --- */
;/* --- 如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序 --- */

#include <reg52.h>
#include <intrins.h>        /* use _nop_() function */

sfr AUXR = 0x8e;
sfr AUXR1 = 0xa2;

sfr P4 = 0xe8;
sfr XICON = 0xc0;

sfr IPH = 0xb7;

sfr WDT_CONTR = 0xe1;
sfr ISP_DATA = 0xe2;
sfr ISP_ADDRH = 0xe3;
sfr ISP_ADDRL = 0xe4;
sfr ISP_CMD = 0xe5;
sfr ISP_TRIG = 0xe6;
sfr ISP_CONTR = 0xe7;

sbit ERROR_LED = P1^5;
sbit OK_LED = P1^7;

void main()
{
    unsigned int array_point = 0;

    /* 测试数组 Test_array_one[512],Test_array_two[512]*/
    unsigned char xdata Test_array_one[512]        =
    {
        0x00,     0x01,     0x02,     0x03,     0x04,     0x05,     0x06,     0x07,
        0x08,     0x09,     0x0a,     0x0b,     0x0c,     0x0d,     0x0e,     0x0f,
        0x10,     0x11,     0x12,     0x13,     0x14,     0x15,     0x16,     0x17,
        0x18,     0x19,     0x1a,     0x1b,     0x1c,     0x1d,     0x1e,     0x1f,
        0x20,     0x21,     0x22,     0x23,     0x24,     0x25,     0x26,     0x27,
```

```
0x28,   0x29,   0x2a,   0x2b,   0x2c,   0x2d,   0x2e,   0x2f,
0x30,   0x31,   0x32,   0x33,   0x34,   0x35,   0x36,   0x37,
0x38,   0x39,   0x3a,   0x3b,   0x3c,   0x3d,   0x3e,   0x3f,
0x40,   0x41,   0x42,   0x43,   0x44,   0x45,   0x46,   0x47,
0x48,   0x49,   0x4a,   0x4b,   0x4c,   0x4d,   0x4e,   0x4f,
0x50,   0x51,   0x52,   0x53,   0x54,   0x55,   0x56,   0x57,
0x58,   0x59,   0x5a,   0x5b,   0x5c,   0x5d,   0x5e,   0x5f,
0x60,   0x61,   0x62,   0x63,   0x64,   0x65,   0x66,   0x67,
0x68,   0x69,   0x6a,   0x6b,   0x6c,   0x6d,   0x6e,   0x6f,
0x70,   0x71,   0x72,   0x73,   0x74,   0x75,   0x76,   0x77,
0x78,   0x79,   0x7a,   0x7b,   0x7c,   0x7d,   0x7e,   0x7f,
0x80,   0x81,   0x82,   0x83,   0x84,   0x85,   0x86,   0x87,
0x88,   0x89,   0x8a,   0x8b,   0x8c,   0x8d,   0x8e,   0x8f,
0x90,   0x91,   0x92,   0x93,   0x94,   0x95,   0x96,   0x97,
0x98,   0x99,   0x9a,   0x9b,   0x9c,   0x9d,   0x9e,   0x9f,
0xa0,   0xa1,   0xa2,   0xa3,   0xa4,   0xa5,   0xa6,   0xa7,
0xa8,   0xa9,   0xaa,   0xab,   0xac,   0xad,   0xae,   0xaf,
0xb0,   0xb1,   0xb2,   0xb3,   0xb4,   0xb5,   0xb6,   0xb7,
0xb8,   0xb9,   0xba,   0xbb,   0xbc,   0xbd,   0xbe,   0xbf,
0xc0,   0xc1,   0xc2,   0xc3,   0xc4,   0xc5,   0xc6,   0xc7,
0xc8,   0xc9,   0xca,   0xcb,   0xcc,   0xcd,   0xce,   0xcf,
0xd0,   0xd1,   0xd2,   0xd3,   0xd4,   0xd5,   0xd6,   0xd7,
0xd8,   0xd9,   0xda,   0xdb,   0xdc,   0xdd,   0xde,   0xdf,
0xe0,   0xe1,   0xe2,   0xe3,   0xe4,   0xe5,   0xe6,   0xe7,
0xe8,   0xe9,   0xea,   0xeb,   0xec,   0xed,   0xee,   0xef,
0xf0,   0xf1,   0xf2,   0xf3,   0xf4,   0xf5,   0xf6,   0xf7,
0xf8,   0xf9,   0xfa,   0xfb,   0xfc,   0xfd,   0xfe,   0xff,
0xff,   0xfe,   0xfd,   0xfc,   0xfb,   0xfa,   0xf9,   0xf8,
0xf7,   0xf6,   0xf5,   0xf4,   0xf3,   0xf2,   0xf1,   0xf0,
0xef,   0xee,   0xed,   0xec,   0xeb,   0xea,   0xe9,   0xe8,
0xe7,   0xe6,   0xe5,   0xe4,   0xe3,   0xe2,   0xe1,   0xe0,
0xdf,   0xde,   0xdd,   0xdc,   0xdb,   0xda,   0xd9,   0xd8,
0xd7,   0xd6,   0xd5,   0xd4,   0xd3,   0xd2,   0xd1,   0xd0,
0xcf,   0xce,   0xcd,   0xcc,   0xcb,   0xca,   0xc9,   0xc8,
0xc7,   0xc6,   0xc5,   0xc4,   0xc3,   0xc2,   0xc1,   0xc0,
0xbf,   0xbe,   0xbd,   0xbc,   0xbb,   0xba,   0xb9,   0xb8,
0xb7,   0xb6,   0xb5,   0xb4,   0xb3,   0xb2,   0xb1,   0xb0,
0xaf,   0xae,   0xad,   0xac,   0xab,   0xaa,   0xa9,   0xa8,
0xa7,   0xa6,   0xa5,   0xa4,   0xa3,   0xa2,   0xa1,   0xa0,
0x9f,   0x9e,   0x9d,   0x9c,   0x9b,   0x9a,   0x99,   0x98,
0x97,   0x96,   0x95,   0x94,   0x93,   0x92,   0x91,   0x90,
0x8f,   0x8e,   0x8d,   0x8c,   0x8b,   0x8a,   0x89,   0x88,
0x87,   0x86,   0x85,   0x84,   0x83,   0x82,   0x81,   0x80,
0x7f,   0x7e,   0x7d,   0x7c,   0x7b,   0x7a,   0x79,   0x78,
0x77,   0x76,   0x75,   0x74,   0x73,   0x72,   0x71,   0x70,
0x6f,   0x6e,   0x6d,   0x6c,   0x6b,   0x6a,   0x69,   0x68,
0x67,   0x66,   0x65,   0x64,   0x63,   0x62,   0x61,   0x60,
0x5f,   0x5e,   0x5d,   0x5c,   0x5b,   0x5a,   0x59,   0x58,
```

```
        0x57,   0x56,   0x55,   0x54,   0x53,   0x52,   0x51,   0x50,
        0x4f,   0x4e,   0x4d,   0x4c,   0x4b,   0x4a,   0x49,   0x48,
        0x47,   0x46,   0x45,   0x44,   0x43,   0x42,   0x41,   0x40,
        0x3f,   0x3e,   0x3d,   0x3c,   0x3b,   0x3a,   0x39,   0x38,
        0x37,   0x36,   0x35,   0x34,   0x33,   0x32,   0x31,   0x30,
        0x2f,   0x2e,   0x2d,   0x2c,   0x2b,   0x2a,   0x29,   0x28,
        0x27,   0x26,   0x25,   0x24,   0x23,   0x22,   0x21,   0x20,
        0x1f,   0x1e,   0x1d,   0x1c,   0x1b,   0x1a,   0x19,   0x18,
        0x17,   0x16,   0x15,   0x14,   0x13,   0x12,   0x11,   0x10,
        0x0f,   0x0e,   0x0d,   0x0c,   0x0b,   0x0a,   0x09,   0x08,
        0x07,   0x06,   0x05,   0x04,   0x03,   0x02,   0x01,   0x00
    };

    unsigned char xdata Test_array_two[512]       =
    {
        0x00,   0x01,   0x02,   0x03,   0x04,   0x05,   0x06,   0x07,
        0x08,   0x09,   0x0a,   0x0b,   0x0c,   0x0d,   0x0e,   0x0f,
        0x10,   0x11,   0x12,   0x13,   0x14,   0x15,   0x16,   0x17,
        0x18,   0x19,   0x1a,   0x1b,   0x1c,   0x1d,   0x1e,   0x1f,
        0x20,   0x21,   0x22,   0x23,   0x24,   0x25,   0x26,   0x27,
        0x28,   0x29,   0x2a,   0x2b,   0x2c,   0x2d,   0x2e,   0x2f,
        0x30,   0x31,   0x32,   0x33,   0x34,   0x35,   0x36,   0x37,
        0x38,   0x39,   0x3a,   0x3b,   0x3c,   0x3d,   0x3e,   0x3f,
        0x40,   0x41,   0x42,   0x43,   0x44,   0x45,   0x46,   0x47,
        0x48,   0x49,   0x4a,   0x4b,   0x4c,   0x4d,   0x4e,   0x4f,
        0x50,   0x51,   0x52,   0x53,   0x54,   0x55,   0x56,   0x57,
        0x58,   0x59,   0x5a,   0x5b,   0x5c,   0x5d,   0x5e,   0x5f,
        0x60,   0x61,   0x62,   0x63,   0x64,   0x65,   0x66,   0x67,
        0x68,   0x69,   0x6a,   0x6b,   0x6c,   0x6d,   0x6e,   0x6f,
        0x70,   0x71,   0x72,   0x73,   0x74,   0x75,   0x76,   0x77,
        0x78,   0x79,   0x7a,   0x7b,   0x7c,   0x7d,   0x7e,   0x7f,
        0x80,   0x81,   0x82,   0x83,   0x84,   0x85,   0x86,   0x87,
        0x88,   0x89,   0x8a,   0x8b,   0x8c,   0x8d,   0x8e,   0x8f,
        0x90,   0x91,   0x92,   0x93,   0x94,   0x95,   0x96,   0x97,
        0x98,   0x99,   0x9a,   0x9b,   0x9c,   0x9d,   0x9e,   0x9f,
        0xa0,   0xa1,   0xa2,   0xa3,   0xa4,   0xa5,   0xa6,   0xa7,
        0xa8,   0xa9,   0xaa,   0xab,   0xac,   0xad,   0xae,   0xaf,
        0xb0,   0xb1,   0xb2,   0xb3,   0xb4,   0xb5,   0xb6,   0xb7,
        0xb8,   0xb9,   0xba,   0xbb,   0xbc,   0xbd,   0xbe,   0xbf,
        0xc0,   0xc1,   0xc2,   0xc3,   0xc4,   0xc5,   0xc6,   0xc7,
        0xc8,   0xc9,   0xca,   0xcb,   0xcc,   0xcd,   0xce,   0xcf,
        0xd0,   0xd1,   0xd2,   0xd3,   0xd4,   0xd5,   0xd6,   0xd7,
        0xd8,   0xd9,   0xda,   0xdb,   0xdc,   0xdd,   0xde,   0xdf,
        0xe0,   0xe1,   0xe2,   0xe3,   0xe4,   0xe5,   0xe6,   0xe7,
        0xe8,   0xe9,   0xea,   0xeb,   0xec,   0xed,   0xee,   0xef,
        0xf0,   0xf1,   0xf2,   0xf3,   0xf4,   0xf5,   0xf6,   0xf7,
        0xf8,   0xf9,   0xfa,   0xfb,   0xfc,   0xfd,   0xfe,   0xff,
        0xff,   0xfe,   0xfd,   0xfc,   0xfb,   0xfa,   0xf9,   0xf8,
```

```
        0xf7,   0xf6,   0xf5,   0xf4,   0xf3,   0xf2,   0xf1,   0xf0,
        0xef,   0xee,   0xed,   0xec,   0xeb,   0xea,   0xe9,   0xe8,
        0xe7,   0xe6,   0xe5,   0xe4,   0xe3,   0xe2,   0xe1,   0xe0,
        0xdf,   0xde,   0xdd,   0xdc,   0xdb,   0xda,   0xd9,   0xd8,
        0xd7,   0xd6,   0xd5,   0xd4,   0xd3,   0xd2,   0xd1,   0xd0,
        0xcf,   0xce,   0xcd,   0xcc,   0xcb,   0xca,   0xc9,   0xc8,
        0xc7,   0xc6,   0xc5,   0xc4,   0xc3,   0xc2,   0xc1,   0xc0,
        0xbf,   0xbe,   0xbd,   0xbc,   0xbb,   0xba,   0xb9,   0xb8,
        0xb7,   0xb6,   0xb5,   0xb4,   0xb3,   0xb2,   0xb1,   0xb0,
        0xaf,   0xae,   0xad,   0xac,   0xab,   0xaa,   0xa9,   0xa8,
        0xa7,   0xa6,   0xa5,   0xa4,   0xa3,   0xa2,   0xa1,   0xa0,
        0x9f,   0x9e,   0x9d,   0x9c,   0x9b,   0x9a,   0x99,   0x98,
        0x97,   0x96,   0x95,   0x94,   0x93,   0x92,   0x91,   0x90,
        0x8f,   0x8e,   0x8d,   0x8c,   0x8b,   0x8a,   0x89,   0x88,
        0x87,   0x86,   0x85,   0x84,   0x83,   0x82,   0x81,   0x80,
        0x7f,   0x7e,   0x7d,   0x7c,   0x7b,   0x7a,   0x79,   0x78,
        0x77,   0x76,   0x75,   0x74,   0x73,   0x72,   0x71,   0x70,
        0x6f,   0x6e,   0x6d,   0x6c,   0x6b,   0x6a,   0x69,   0x68,
        0x67,   0x66,   0x65,   0x64,   0x63,   0x62,   0x61,   0x60,
        0x5f,   0x5e,   0x5d,   0x5c,   0x5b,   0x5a,   0x59,   0x58,
        0x57,   0x56,   0x55,   0x54,   0x53,   0x52,   0x51,   0x50,
        0x4f,   0x4e,   0x4d,   0x4c,   0x4b,   0x4a,   0x49,   0x48,
        0x47,   0x46,   0x45,   0x44,   0x43,   0x42,   0x41,   0x40,
        0x3f,   0x3e,   0x3d,   0x3c,   0x3b,   0x3a,   0x39,   0x38,
        0x37,   0x36,   0x35,   0x34,   0x33,   0x32,   0x31,   0x30,
        0x2f,   0x2e,   0x2d,   0x2c,   0x2b,   0x2a,   0x29,   0x28,
        0x27,   0x26,   0x25,   0x24,   0x23,   0x22,   0x21,   0x20,
        0x1f,   0x1e,   0x1d,   0x1c,   0x1b,   0x1a,   0x19,   0x18,
        0x17,   0x16,   0x15,   0x14,   0x13,   0x12,   0x11,   0x10,
        0x0f,   0x0e,   0x0d,   0x0c,   0x0b,   0x0a,   0x09,   0x08,
        0x07,   0x06,   0x05,   0x04,   0x03,   0x02,   0x01,   0x00
    };
    ERROR_LED = 1;
    OK_LED = 1;
    for(array_point=0; array_point<512; array_point++)
    {
        if(Test_array_one[array_point]!=Test_array_two [array_point]){
            ERROR_LED = 0;
            OK_LED = 1;
            break;
        }
        else{
            OK_LED = 0;
            ERROR_LED = 1;
        }
    }
    while(1);
}
```

双数据指针 DPTR0,DPTR1 的使用

适用型号：

STC89C51RC,STC89C52RC,STC89C53RC,STC89LE51RC,STC89LE52RC,STC89LE53RC
STC89C54RD+,STC89C58RD+,STC89C516RD+,STC89LE54RD+,STC89LE58RD+,STC89LE516RD+
STC89LE516AD,STC89LE516X2

RC/RD+/AD/X2 系列 8051 单片机 双数据指针 特殊功能寄存器

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
AUXR1	A2h	Auxiliary Register 1	-	-	-	-	GF2	-	-	DPS	xxxx,0xx0

Symbol 符号 Function 功能

GF2 General purpose user-defined flag. 通用功能用户自定义位

DPS DPTR registers select bit. DPTR 寄存器选择位

0: DPTR0 is selected DPTR0 被选择

1: DPTR1 is selected DPTR1 被选择

此系列单片机有两个 16-bit 数据指针,DPTR0,DPTR1.当 DPS 选择位为 0 时,选择 DPTR0, 当 DPS 选择位为 1 时,选择 DPTR1.

AUXR1 特殊功能寄存器,位于 A2H 单元,其中的位不可用布尔指令快速访问.但由于 DPS 位位于 bit0,故对 AUXR1 寄存器用 INC 指令,DPS 位便会反转,由 0 变成 1 或由 1 变成 0,即可实现双数据指针的快速切换.

应用示例供参考：

```
;新增特殊功能寄存器定义
AUXR1   DATA    0A2H
MOV     AUXR1,  #0      ;此时 DPS 为 0,DPTR0 有效

MOV     DPTR,   #1FFH   ;置 DPTR0 为 1FFH
MOV     A,      #55H
MOVX    @DPTR,  A       ;将 1FFH 单元置为 55H

MOV     DPTR,   #2FFH   ; 置 DPTR0 为 2FFH
MOV     A,      #0AAH
MOVX    @DPTR,  A       ;将 2FFH 单元置为 0AAH

INC     AUXR1           ; 此时 DPS 为 1,DPTR1 有效
MOV     DPTR,   #1FFH   ; 置 DPTR1 为 1FFH
MOVX    A,      @DPTR   ; 读 DPTR1 数据指针指向的 1FFH 单元的内容,累加器 A 变为 55H.

INC     AUXR1           ; 此时 DPS 为 0,DPTR0 有效
MOVX    A,      @DPTR   ; 读 DPTR0 数据指针指向的 2FFH 单元的内容,累加器 A 变为 0AAH.

INC     AUXR1           ; 此时 DPS 为 1,DPTR1 有效
MOVX    A,      @DPTR   ; 读 DPTR1 数据指针指向的 1FFH 单元的内容,累加器 A 变为 55H.

INC     AUXR1           ; 此时 DPS 为 0,DPTR0 有效
MOVX    A,      @DPTR   ; 读 DPTR0 数据指针指向的 2FFH 单元的内容,累加器 A 变为 0AAH.
```

P4 口 （可以位寻址，可像操作 P1/P2/P3 一样操作 P4 口）

STC89C51RC/RD+系列 8051 单片机 I/0 口 特殊功能寄存器 Port SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P0	80h	8-bit Port 0	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	1111,1111
P1	90h	8-bit Port 1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	1111,1111
P2	A0h	8-bit Port 2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	1111,1111
P3	B0h	8-bit Port 3	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	1111,1111
P4	E8h	4-bit Port 4	-	-	-	-	P4.3	P4.2	P4.1	P4.0	xxxx,1111

汇编语言：

```
P4      DATA    0E8H ;  or  P4  EQU  0E8H
MOV     A,      P4   ;  Read P4 status to Accumulator.
MOV     P4,     #0AH ;  Output data “A”through P4.0 - P4.3
SETB    P4.0         ;  P4.0 = 1
CLR     P4.0         ;  P4.0 = 0
MOV     P4,     #0AH ;  Output data “A”through P4.0 - P4.3
```

C 语言：

```
sfr     P4   =   0xe8;C 语言中声明 P4 口特殊功能寄存器地址
```

注：STC89LE516AD,STC89LE516X2,STC89LE58AD,STC89LE54AD 的 P4 口地址在 C0h。

看门狗应用

适用型号:

STC89C51RC,STC89C52RC,STC89C53RC,STC89LE51RC,STC89LE52RC,STC89LE53RC
STC89C54RD+,STC89C58RD+,STC89C516RD+,STC89LE54RD+,STC89LE58RD+,STC89LE516RD+

STC89C51RC/RD+系列 8051单片机 看门狗定时器 特殊功能寄存器 Watch Dog Timer SFR

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
WDT_CONTR	E1h	Watch-Dog-Timer Control register	-	-	EN_WDT	CLR_WDT	IDLE_WDT	PS2	PS1	PS0	xx00,0000

Symbol 符号 Function 功能

EN_WDT Enable WDT bit. When set, WDT is started
看门狗允许位，当设置为“1”时，看门狗启动。

CLR_WDT WDT clear bit. When set, WDT will recount. Hardware will automatically clear this bit.
看门狗清“0”位,当设为“1”时，看门狗将重新计数。硬件将自动清“0”此位。

IDLE_WDT When set, WDT is enabled in IDLE mode. When clear, WDT is disabled in IDLE mode
看门狗“IDLE”模式位,当设置为“1”时,看门狗定时器在“空闲模式”计数
当清“0”该位时，看门狗定时器在“空闲模式”时不计数

PS2, PS1, PS0 Pre-scale value of Watchdog timer is shown as the bellowed table:
看门狗定时器预分频值，如下表所示

PS2	PS1	PS0	Pre-scale 预分频	WDT Period @20MHz and 12 clocks mode
0	0	0	2	39.3 mS
0	0	1	4	78.6 mS
0	1	0	8	157.3 mS
0	1	1	16	314.6 mS
1	0	0	32	629.1 mS
1	0	1	64	1.25S
1	1	0	128	2.5S
1	1	1	256	5S

The WDT period is determined by the following equation 看门狗溢出时间计算
看门狗溢出时间= (N x Pre-scale x 32768) / Oscillator frequency
N = 12, 当在12 clock mode时, N = 6, 当在6 clock mode 时
设时钟为12MHz，12时钟模式
看门狗溢出时间 = (12 x Pre-scale x 32768) / 12000000 = Pre-scale x 393216 / 12000000

PS2	PS1	PS0	Pre-scale 预分频	WDT Period @12MHz and 12 clocks mode
0	0	0	2	65.5 mS
0	0	1	4	131.0 mS
0	1	0	8	262.1 mS
0	1	1	16	524.2 mS
1	0	0	32	1.0485S
1	0	1	64	2.0971S
1	1	0	128	4.1943S
1	1	1	256	8.3886S

设时钟为11.0592MHz，12时钟模式

看门狗溢出时间 = (12 x Pre-scale x 32768) / 11059200 = Pre-scale x 393216 / 11059200

PS2	PS1	PS0	Pre-scale 预分频	WDT Period @11.0592MHz and 12 clocks mode
0	0	0	2	71.1 mS
0	0	1	4	142.2 mS
0	1	0	8	284.4 mS
0	1	1	16	568.8 mS
1	0	0	32	1.1377S
1	0	1	64	2.2755S
1	1	0	128	4.5511S
1	1	1	256	9.1022S

汇编语言程序示例

```
WDT_CONTR   DATA    0E1H ;     或者    WDT_CONTR  EQU    0E1H
;复位入口
    ORG     0000H
    LJMP    Initial
    …
    ORG     0060H
Initial:
    MOV     WDT_CONTR, #00111100B; Load initial value 看门狗定时器控制寄存器初始化
                ; EN_WDT = 1, CLR_WDT = 1, IDLE_WDT = 1, PS2 = 1, PS1 = 0, PS0 = 0
    …
Main_Loop:
    LCALL   Display_Loop
    LCALL   Keyboard_Loop
    …
    MOV     WDT_CONTR, #00111100B ;  喂狗, 不要用 ORL    WDT_CONTR, #00010000B
    …
    LJMP    Main_Loop
```

C语言程序示例

```
#include<reg52.h>
sfr     WDT_CONTR   =   0xe1;
void main()
{   …
    WDT_CONTR   =   0x3c;
    /* 0011,0100 EN_WDT = 1,CLR_WDT = 1, IDLE_WDT = 1, PS2 = 1, PS1 = 0, PS0 = 0 */
    while(1){
        display();
        keyboard();
        …
        WDT_CONTR   =   0x3c;  /* 喂狗, 不要用 WDT_CONTR  =  WDT_CONTR | 0x10;*/
    }
}
```

```
;/* ---  STC International Limited -------------------------------------- */
;/* ---  宏晶科技 姚永平  设计 2006/1/6   V1.0 --------------------------- */
;/* ---  演示 STC89C/LE51RC/RD+ 系列 MCU 看门狗及其溢出时间计算公式--------- */
;/* ---  Mobile: 13922805190 ------------------------------------------- */
;/* ---  Fax: 0755-82944243 -------------------------------------------- */
;/* ---  Tel: 0755-82948409 -------------------------------------------- */
;/* ---  Web: www.mcu-memory.com --------------------------------------- */
;/* ---  本演示程序在STC-ISP Ver 3.0A.PCB的下载编程工具上测试通过 ---------- */
;/* ---  如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 --- */
;/* ---  如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序 --- */

;本程序用于验证 STC89C/LE51RC/RD+ 系列单片机的看门狗及其溢出时间计算公式
;看门狗及其溢出时间 = (N * Pre_scale *32768)/Oscillator frequency
;  N = 12, 当在 12 clock mode 时, N = 6, 当在 6 clock mode 时。

WDTCR       EQU      0E1H          ;看门狗地址
LED         EQU      P1.5          ;用 P1.5 控制发光二级管

Pre_scale_Word  EQU   0x35      ;清0、启动看门狗,预分频数=64
;f=18.432MHz、12clock mode 时
;    看门狗溢出时间 = (12 * 64 * 32768)/18432000 = 1.36S

    ORG   0000H
    AJMP  MAIN

    ORG   0100H
MAIN:
    CLR   LED                     ;点亮 LED
    ACALL Delay                   ;延时,让 LED 亮大约 1S 的时间

    MOV   WDTCR, #Pre_scale_Word ;启动看门狗,若注释掉本条指令即不启动狗,
                                   ;LED 只会亮一次
    SETB  LED                      ;熄灭 LED
Wait:
    SJMP  Wait          ;跳转到本语句(停机),等待看门狗溢出复位, 复位后将再次点亮 LED

Delay:
    MOV   R0, #0
    MOV   R1, #0
    MOV   R2, #15
Delay_Loop:
    DJNZ  R0, Delay_Loop
    DJNZ  R1, Delay_Loop
    DJNZ  R2, Delay_Loop
    RET
    END
```

STC89C51RC/RD+系列单片机如何用软件实现系统复位

用户应用程序在运行过程当中，有时会有特殊需求，需要实现单片机系统软复位（热启动之一），传统的8051单片机由于硬件上未支持此功能，用户必须用软件模拟实现，实现起来较麻烦。现STC新推出的增强型8051根据客户要求增加了ISP_CONTR 特殊功能寄存器,实现了此功能。用户只需简单的控制ISP_CONTR 特殊功能寄存器的其中两位 SWBS / SWRST 就可以系统复位了。

ISP_CONTR：ISP/IAP控制寄存器，地址在 0E7H单元

B7	B6	B5	B4	B3	B2	B1	B0	Reset Value
ISPEN	SWBS	SWRST	-	-	WT2	WT1	WT0	000x,x000

ISPEN: ISP/IAP 功能允许位。0：禁止ISP/IAP编程改变Flash,1:允许编程改变Flash
SWBS: 软件选择从用户应用程序区启动（0），还是从ISP程序区启动（1）。要与SWRST直接配合才可以实现
SWRST: 0: 不操作；1: 产生软件系统复位，硬件自动清零。

```
;从用户应用程序区(AP区)软件复位并切换到用户应用程序区(AP区)开始执行程序
    MOV   ISP_CONTR,#00100000B ;SWBS = 0(选择AP区), SWRST = 1(软复位)
;从系统ISP监控程序区软件复位并切换到用户应用程序区(AP区)开始执行程序
    MOV   ISP_CONTR,#00100000B ;SWBS = 0(选择AP区), SWRST = 1(软复位)
;从用户应用程序区(AP区)软件复位并切换到系统ISP监控程序区开始执行程序
    MOV   ISP_CONTR,#01100000B ;SWBS = 1(选择ISP区), SWRST = 1(软复位)
;从系统ISP监控程序区软件复位并切换到系统ISP监控程序区开始执行程序
    MOV   ISP_CONTR,#01100000B ;SWBS = 1(选择ISP区), SWRST = 1(软复位)
```

本复位是整个系统复位，所有的特殊功能寄存器都会复位到初始值，I/O口也会初始化。

STC89C51 RC / RD+ 系列 ISP / IAP 应用

STC89C51 RC / RD+ 系列 内部EEPROM的应用

-- 利用IAP技术可实现EEPROM，内部Flash擦写次数为100,000次以上

RC/RD+系列 8051 单片机 ISP/IAP 特殊功能寄存器　　ISP/IAP SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
ISP_DATA	E2h	ISP/IAP Flash Data Register									1111,1111
ISP_ADDRH	E3h	ISP/IAP Flash Address High									0000,0000
ISP_ADDRL	E4h	ISP/IAP Flash Address Low									0000,0000
ISP_CMD	E5h	ISP/IAP Flash Command Register	-	-	-	-	-	MS2	MS1	MS0	xxxx,x000
ISP_TRIG	E6h	ISP/IAP Flash Command Trigger									xxxx,xxxx
ISP_CONTR	E7h	ISP/IAP Control Register	ISPEN	SWBS	SWRST	-	-	WT2	WT1	WT0	000x,x000

ISP_DATA:　　　ISP/IAP操作时的数据寄存器。

　　　　　　　ISP/IAP从Flash读出的数据放在此处，向Flash写的数据也需放在此处

ISP_ADDRH: ISP/IAP操作时的地址寄存器高八位。

ISP_ADDRL: ISP/IAP操作时的地址寄存器低八位。

ISP_CMD:　　　ISP/IAP操作时的命令模式寄存器，须命令触发寄存器触发方可生效。

B7	B6	B5	B4	B3	B2	B1	B0	命令 / 操作 模式选择
保留					命令 选择			
-	-	-	-	-	0	0	0	Standby　待机模式，无ISP操作
-	-	-	-	-	0	0	1	AP-Flash / Data-Flash Read 对用户的应用程序Flash区及数据Flash区字节读
-	-	-	-	-	0	1	0	AP-Flash / Data-Flash Program 对用户的应用程序Flash区及数据Flash区字节编程
-	-	-	-	-	0	1	1	AP-Flash / Data-Flash Sector Erase 对用户的应用程序Flash区及数据Flash区扇区擦除

程序在系统ISP程序区时可以对用户应用程序区 / 数据Flash区(EEPROM)进行字节读 / 字节编程 / 扇区擦除；程序在用户应用程序区时，仅可以对数据Flash区(EEPROM)进行字节读 / 字节编程 / 扇区擦除。已经固化有ISP引导码,并设置为上电复位进入ISP的STC89C51RC/RD+系列单片机出厂时就已完全加密。

ISP_TRIG:　　　ISP/IAP操作时的命令触发寄存器。

　　　　　　　在ISPEN(ISP_CONTR.7) = 1时,对ISP_TRIG先写入46h,再写入B9h,

　　　　　　　ISP/IAP命令才会生效。

ISP_CONTR: ISP/IAP控制寄存器。

B7	B6	B5	B4	B3	B2	B1	B0	Reset Value
ISPEN	SWBS	SWRST	-	-	WT2	WT1	WT0	000x,x000

ISPEN:　　ISP/IAP 功能允许位。0：禁止ISP/IAP编程改变Flash,1:允许编程改变Flash

SWBS:软件选择从用户主程序区启动（0），还是从ISP 程序区启动（1）。

SWRST:　　0: 不操作；1: 产生软件系统复位，硬件自动清零。

;从用户应用程序区(AP区)软件复位并切换到ISP程序区开始执行程序

　　MOV　ISP_CONTR, #01100000B ;SWBS = 1(选择ISP区), SWRST = 1(软复位)

;从ISP程序区软件复位并切换到用户应用程序区(AP区)开始执行程序

　　MOV　ISP_CONTR, #00100000B ;SWBS = 0(选择AP区), SWRST = 1(软复位)

设置等待时间			CPU 等待时间(机器周期)			
WT2	WT1	WT0	Read	Program	Sector Erase	Recommended System Clock
0	1	1	6	30	5471	5MHz
0	1	0	11	60	10942	10MHz
0	0	1	22	120	21885	20MHz
0	0	0	43	240	43769	40MHz

STC89C51RC,STC89LE51RC 单片机内部可用 Data Flash(EEPROM)的地址：

第一扇区		第二扇区		第三扇区		第四扇区		每个扇区512字节 建议同一次修改的数据放在同一扇区
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
2000h	21FFh	2200h	23FFh	2400h	25FFh	2600h	27FFh	
第五扇区		第六扇区		第七扇区		第八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
2800h	29FFh	2A00h	2BFFh	2C00h	2DFFh	2E00h	2FFFh	

STC89C52RC,STC89LE52RC 单片机内部可用 Data Flash(EEPROM)的地址：

第一扇区		第二扇区		第三扇区		第四扇区		每个扇区512字节 建议同一次修改的数据放在同一扇区
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
2000h	21FFh	2200h	23FFh	2400h	25FFh	2600h	27FFh	
第五扇区		第六扇区		第七扇区		第八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
2800h	29FFh	2A00h	2BFFh	2C00h	2DFFh	2E00h	2FFFh	

STC89C54RD+,STC89LE54RD+单片机内部可用Data Flash(EEPROM)的地址：

第一扇区		第二扇区		第三扇区		第四扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
8000h	81FFh	8200h	83FFh	8400h	85FFh	8600h	87FFh
第五扇区		第六扇区		第七扇区		第八扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
8800h	89FFh	8A00h	8BFFh	8C00h	8DFFh	8E00h	8FFFh
第九扇区		第十扇区		第十一扇区		第十二扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
9000h	91FFh	9200h	93FFh	9400h	95FFh	9600h	97FFh
第十三扇区		第十四扇区		第十五扇区		第十六扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
9800h	99FFh	9A00h	9BFFh	9C00h	9DFFh	9E00h	9FFFh
第十七扇区		第十八扇区		第十九扇区		第二十扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
A000h	A1FFh	A200h	A3FFh	A400h	A5FFh	A600h	A7FFh
第二十一扇区		第二十二扇区		第二十三扇区		第二十四扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
A800h	A9FFh	AA00h	ABFFh	AC00h	ADFFh	AE00h	AFFFH
第二十五扇区		第二十六扇区		第二十七扇区		第二十八扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
B000h	B1FFh	B200h	B3FFh	B400h	B5FFh	B600h	B7FFH
第二十九扇区		第三十扇区		第三十一扇区		第三十二扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
B800h	B9FFh	BA00h	BBFFh	BC00h	BDFFh	BE00h	BFFFH
第三十三扇区		第三十四扇区		第三十五扇区		第三十六扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
C000h	C1FFh	C200h	C3FFh	C400h	C5FFh	C600h	C7FFH
第三十七扇区		第三十八扇区		第三十九扇区		第四十扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
C800h	C9FFh	CA00h	CBFFh	CC00h	CDFFh	CE00h	CFFFH
第四十一扇区		第四十二扇区		第四十三扇区		第四十四扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
D000h	D1FFh	D200h	D3FFh	D400h	D5FFh	D600h	D7FFH
第四十五扇区		第四十六扇区		第四十七扇区		第四十八扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
D800h	D9FFh	DA00h	DBFFh	DC00h	DDFFh	DE00h	DFFFH
第四十九扇区		第五十扇区		第五十一扇区		第五十二扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
E000h	E1FFh	E200h	E3FFh	E400h	E5FFh	E600h	E7FFH
第五十三扇区		第五十四扇区		第五十五扇区		第五十六扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
E800h	E9FFh	EA00h	EBFFh	EC00h	EDFFh	EE00h	EFFFH
第五十七扇区		第五十八扇区					
起始地址	结束地址	起始地址	结束地址				
F000h	F1FFh	F200h	F3FFh				

每个扇区512字节

建议同一次修改的数据放在同一扇区

STC89C55RD+,STC89LE55RD+ 单片机内部可用 Data Flash(EEPROM)的地址：

第一扇区		第二扇区		第三扇区		第四扇区		每个扇区512字节 建议同一次修改的数据放在同一扇区
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
8000h	81FFh	8200h	83FFh	8400h	85FFh	8600h	87FFh	
第五扇区		第六扇区		第七扇区		第八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
8800h	89FFh	8A00h	8BFFh	8C00h	8DFFh	8E00h	8FFFh	
第九扇区		第十扇区		第十一扇区		第十二扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
9000h	91FFh	9200h	93FFh	9400h	95FFh	9600h	97FFh	
第十三扇区		第十四扇区		第十五扇区		第十六扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
9800h	99FFh	9A00h	9BFFh	9C00h	9DFFh	9E00h	9FFFh	
第十七扇区		第十八扇区		第十九扇区		第二十扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
A000h	A1FFh	A200h	A3FFh	A400h	A5FFh	A600h	A7FFh	
第二十一扇区		第二十二扇区		第二十三扇区		第二十四扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
A800h	A9FFh	AA00h	ABFFh	AC00h	ADFFh	AE00h	AFFFH	
第二十五扇区		第二十六扇区		第二十七扇区		第二十八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
B000h	B1FFh	B200h	B3FFh	B400h	B5FFh	B600h	B7FFH	
第二十九扇区		第三十扇区		第三十一扇区		第三十二扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
B800h	B9FFh	BA00h	BBFFh	BC00h	BDFFh	BE00h	BFFFH	
第三十三扇区		第三十四扇区		第三十五扇区		第三十六扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
C000h	C1FFh	C200h	C3FFh	C400h	C5FFh	C600h	C7FFH	
第三十七扇区		第三十八扇区		第三十九扇区		第四十扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
C800h	C9FFh	CA00h	CBFFh	CC00h	CDFFh	CE00h	CFFFH	
第四十一扇区		第四十二扇区		第四十三扇区		第四十四扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
D000h	D1FFh	D200h	D3FFh	D400h	D5FFh	D600h	D7FFH	
第四十五扇区		第四十六扇区		第四十七扇区		第四十八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
D800h	D9FFh	DA00h	DBFFh	DC00h	DDFFh	DE00h	DFFFH	
第四十九扇区		第五十扇区		第五十一扇区		第五十二扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
E000h	E1FFh	E200h	E3FFh	E400h	E5FFh	E600h	E7FFH	
第五十三扇区		第五十四扇区		第五十五扇区		第五十六扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
E800h	E9FFh	EA00h	EBFFh	EC00h	EDFFh	EE00h	EFFFH	
第五十七扇区		第五十八扇区						
起始地址	结束地址	起始地址	结束地址					
F000h	F1FFh	F200h	F3FFh					

STC89C58RD+,STC89LE58RD+单片机内部可用Data Flash(EEPROM)的地址：

第一扇区		第二扇区		第三扇区		第四扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
8000h	81FFh	8200h	83FFh	8400h	85FFh	8600h	87FFh
第五扇区		第六扇区		第七扇区		第八扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
8800h	89FFh	8A00h	8BFFh	8C00h	8DFFh	8E00h	8FFFh
第九扇区		第十扇区		第十一扇区		第十二扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
9000h	91FFh	9200h	93FFh	9400h	95FFh	9600h	97FFh
第十三扇区		第十四扇区		第十五扇区		第十六扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
9800h	99FFh	9A00h	9BFFh	9C00h	9DFFh	9E00h	9FFFh
第十七扇区		第十八扇区		第十九扇区		第二十扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
A000h	A1FFh	A200h	A3FFh	A400h	A5FFh	A600h	A7FFh
第二十一扇区		第二十二扇区		第二十三扇区		第二十四扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
A800h	A9FFh	AA00h	ABFFh	AC00h	ADFFh	AE00h	AFFFH
第二十五扇区		第二十六扇区		第二十七扇区		第二十八扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
B000h	B1FFh	B200h	B3FFh	B400h	B5FFh	B600h	B7FFH
第二十九扇区		第三十扇区		第三十一扇区		第三十二扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
B800h	B9FFh	BA00h	BBFFh	BC00h	BDFFh	BE00h	BFFFH
第三十三扇区		第三十四扇区		第三十五扇区		第三十六扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
C000h	C1FFh	C200h	C3FFh	C400h	C5FFh	C600h	C7FFH
第三十七扇区		第三十八扇区		第三十九扇区		第四十扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
C800h	C9FFh	CA00h	CBFFh	CC00h	CDFFh	CE00h	CFFFH
第四十一扇区		第四十二扇区		第四十三扇区		第四十四扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
D000h	D1FFh	D200h	D3FFh	D400h	D5FFh	D600h	D7FFH
第四十五扇区		第四十六扇区		第四十七扇区		第四十八扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
D800h	D9FFh	DA00h	DBFFh	DC00h	DDFFh	DE00h	DFFFH
第四十九扇区		第五十扇区		第五十一扇区		第五十二扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
E000h	E1FFh	E200h	E3FFh	E400h	E5FFh	E600h	E7FFH
第五十三扇区		第五十四扇区		第五十五扇区		第五十六扇区	
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址
E800h	E9FFh	EA00h	EBFFh	EC00h	EDFFh	EE00h	EFFFH
第五十七扇区		第五十八扇区					
起始地址	结束地址	起始地址	结束地址				
F000h	F1FFh	F200h	F3FFh				

每个扇区512字节

建议同一次修改的数据放在同一扇区

STC89C51 RC / RD+ 系列 IAP 应用汇编简介
STC89C51 RC / RD+ 系列 内部 EEPROM 的应用

```
;用 DATA 还是 EQU 声明新增特殊功能寄存器地址要看你用的汇编器 / 编译器
    ISP_DATA        DATA    0E2h;或  ISP_DATA        EQU     0E2h
    ISP_ADDRH       DATA    0E3h;或  ISP_ADDRH       EQU     0E3h
    ISP_ADDRL       DATA    0E4h;或  ISP_ADDRL       EQU     0E4h
    ISP_CMD         DATA    0E5h;或  ISP_CMD         EQU     0E5h
    ISP_TRIG        DATA    0E6h;或  ISP_TRIG        EQU     0E6h
    ISP_CONTR       DATA    0E7h;或  ISP_CONTR       EQU     0E7h
;定义 ISP/IAP 命令及等待时间
    ISP_IAP_BYTE_READ           EQU   1     ;字节读
    ISP_IAP_BYTE_PROGRAM  EQU   2     ;字节编程,前提是该字节是空，0FFh
    ISP_IAP_SECTOR_ERASE  EQU   3     ;扇区擦除,要某字节为空，要擦一扇区
    WAIT_TIME       EQU   0           ;设置等待时间,40MHz 以下 0,20M 以下 1,
                                      ;             10MHz 以下 2,5M 以下 3
;字节读
    MOV   ISP_ADDRH,#BYTE_ADDR_HIGH;送地址高字节   ┐
    MOV   ISP_ADDRL,#BYTE_ADDR_LOW  ;送地址低字节  ┘ 地址需要改变时才需重新送地址
    CLR   EA    ;关中断,此时各中断请求，会被挂起，一开中断，立即响应,最新的 D 版本不需要关中断
    MOV   ISP_CONTR,#WAIT_TIME  ;设置等待时间        ┐ 此两句可以合成一句,
    ORL   ISP_CONTR,#10000000B  ;允许 ISP/IAP 操作   ┘ 并且只送一次就够了
    MOV   ISP_CMD,   #ISP_IAP_BYTE_READ  ;送字节读命令,命令不需改变时,不需重新送命令
    MOV   ISP_TRIG,  #46h        ;先送 46h,再送 B9h 到 ISP/IAP 触发寄存器,每次都需如此
    MOV   ISP_TRIG,  #0B9h       ;送完 B9h 后，ISP/IAP 命令立即被触发起动
;CPU 等待 IAP 动作完成后，才会继续执行程序，要先关中断（EA),
;再送 46h,B9h 到 ISP/IAP 触发寄存器,起动 ISP/IAP 命令,关中断在触发之前即可
    NOP                              ;数据读出到 ISP_DATA 寄存器后，CPU 继续执行程序
    MOV   A,ISP_DATA                 ;将读出的数据送往 Acc
;以下语句可不用,只是出于安全考虑而已
    MOV   ISP_CONTR,#00000000B        ;禁止 ISP/IAP 操作
    MOV   ISP_CMD,   #00000000B       ;去除 ISP/IAP 命令
    ;MOV  ISP_TRIG,  #00000000B       ;防止 ISP/IAP 命令误触发
    ;MOV  ISP_ADDRH,#0               ;送地址高字节单元为 00,指向非 EEPROM 区
    ;MOV  ISP_ADDRL,#0               ;送地址低字节单元为 00,防止误操作
    SETB  EA                          ;开中断，CPU 处理完 ISP/IAP 动作即可开中断
```

```
;字节编程，该字节为FFh/空时，可对其编程，否则不行,要先执行扇区擦除
        MOV     ISP_DATA,       #ONE_DATA           ;       送字节编程数据到ISP_DATA
        MOV     ISP_ADDRH, #BYTE_ADDR_HIGH   ;送地址高字节 ┐
        MOV     ISP_ADDRL, #BYTE_ADDR_LOW    ;送地址低字节 ┘ 地址需要改变时才需重新送地址
        CLR     EA      ;关中断,此时各中断请求，会被挂起，一开中断，立即响应，最新的D版本不需要关中断
        MOV     ISP_CONTR, #WAIT_TIME  ;设置等待时间     ┐ 此两句可以合成一句，
        ORL     ISP_CONTR, #10000000B  ;允许ISP/IAP操作  ┘ 并且只送一次就够了
        MOV     ISP_CMD,   #ISP_IAP_BYTE_PROGRAM      ;送字节编程命令
        MOV     ISP_TRIG,  #46h        ;先送46h,再送B9h到ISP/IAP触发寄存器,每次都需如此
        MOV     ISP_TRIG,  #0B9h       ;送完B9h后，ISP/IAP命令立即被触发起动
;CPU等待IAP动作完成后，才会继续执行程序，要先关中断（EA），
;再送46h,B9h到ISP/IAP触发寄存器,起动ISP/IAP命令,关中断在触发之前即可
        NOP                                 ;字节编程成功后，CPU继续执行程序
;以下语句可不用,只是出于安全考虑而已
        MOV     ISP_CONTR, #00000000B       ;禁止ISP/IAP操作
        MOV     ISP_CMD,   #00000000B       ;去除ISP/IAP命令
        ;MOV    ISP_TRIG,  #00000000B       ;防止ISP/IAP命令误触发
        ;MOV    ISP_ADDRH, #0               ;送地址高字节单元为00,指向非EEPROM区,防止误操作
        ;MOV    ISP_ADDRL, #0               ;送地址低字节单元为00,指向非EEPROM区,防止误操作
        SETB    EA                          ;开中断，CPU处理完ISP/IAP动作即可开中断
;扇区擦除，没有字节擦除，只有扇区擦除，512字节/扇区，
;扇区里面任意一个字节的地址都是扇区地址,无需求首地址,单片机会自己处理
;建议同一次修改的数据放在同一个扇区
;如果要对某个扇区进行擦除，而其中有些字节的内容需要保留，则需将其先读到单片机
;内部的RAM中保存，再将该扇区擦除，然后将须保留的数据写回该扇区，
;所以每个扇区中用的字节数越少越好，操作起来越灵活越快
;强烈建议同一次修改的数据放在同一个扇区
        MOV     ISP_ADDRH, #SECTOR_FIRST_BYTE_ADDR_HIGH  ;送扇区起始地址高字节 ┐ 地址需要改变时
        MOV     ISP_ADDRL, #SECTOR_FIRST_BYTE_ADDR_LOW   ;送扇区起始地址低字节 ┘ 才需重新送地址
        CLR     EA   ;关中断,此时各中断请求，会被挂起，一开中断，立即响应，最新的D版本不需要关中断
        MOV     ISP_CONTR, #WAIT_TIME  ;设置等待时间 ┐ 此两句可以合成一句，
        ORL     ISP_CONTR, #10000000B  ;允许ISP/IAP  ┘ 并且只送一次就够了
        MOV     ISP_CMD,   #ISP_IAP_SECTOR_ERASE  ;送扇区擦除命令,命令不需改变时,不需重新送命令
        MOV     ISP_TRIG,  #46h        ;先送46h,再送B9h到ISP/IAP触发寄存器,每次都需如此
        MOV     ISP_TRIG,  #0B9h       ;送完B9h后，ISP/IAP命令立即被触发起动
;CPU等待IAP动作完成后，才会继续执行程序，要先关中断（EA），
;再送46h,B9h到ISP/IAP触发寄存器,起动ISP/IAP命令,关中断在触发之前即可
        NOP                                 ;扇区擦除成功后，CPU继续执行程序
;以下语句可不用,只是出于安全考虑而已
        MOV     ISP_CONTR, #00000000B       ;禁止ISP/IAP操作
        MOV     ISP_CMD,   #00000000B       ;去除ISP/IAP命令
        ;MOV    ISP_TRIG,  #00000000B       ;防止ISP/IAP命令误触发
        ;MOV    ISP_ADDRH, #0               ;送地址高字节单元为00,指向非EEPROM区
        ;MOV    ISP_ADDRL, #0               ;送地址低字节单元为00,防止误操作
```

小常识： （STC单片机的Data Flash 当EEPROM功能使用）
3个基本命令----字节读，字节编程，扇区擦除
字节编程：如果该字节是“1111,1111B”,则可将其中的“1”编程为“0”，如果该字节中有位为“0”，则须先将整个扇区擦除，因为只有“扇区擦除”才可以将“0”变为“1”。
扇区擦除：只有“扇区擦除”才可能将“0”擦除为“1”。

大建议：
1.同一次修改的数据放在同一扇区中，单独修改的数据放在另外的扇区,就不须读出保护。
2.如果一个扇区只用一个字节，那就是真正的EEPROM,STC单片机的Data Flash比外部EEPROM要快很多，读一个字节/编程一个字节/擦除一个扇区大概是10uS/60uS/10mS。
3.如果同一个扇区中存放了一个以上的字节，某次只需要修改其中的一个字节或一部分字节时，则另外的不需要修改的数据须先读出放在STC 单片机的RAM中，然后擦除整个扇区，再将需要保留的数据和需修改的数据一并写回该扇区中。这时每个扇区使用的字节数是使用的越少越方便(不需读出一大堆需保留数据)。

```
;/* ---  STC International Limited ---------------------------------------- */
;/* ---  宏晶科技 姚永平   设计 2006/1/6    V1.0 ----------------------------- */
;/* ---  演示 STC89C/LE51RC/RD+ 系列 MCU EEPROM/IAP功能示例程序------------ */
;/* ---  Mobile: 13922805190 --------------------------------------------- */
;/* ---  Tel: 0755-82948409    Fax: 0755-82944243------------------------ */
;/* ---  Web: www.mcu-memory.com ----------------------------------------- */
;/* ---  本演示程序在STC-ISP Ver 3.0A.PCB的下载编程工具上测试通过 ---------- */
;/* ---  如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 --- */
;/* ---  如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序 --- */
;本程序演示STC89C51RC/RD+系列单片机EEPROM/IAP功能
;-------------------------------------------------
;定义与IAP有关的特殊功能寄存器
ISP_DATA        EQU 0E2H
ISP_ADDRH       EQU 0E3H
ISP_ADDRL       EQU 0E4H
ISP_CMD         EQU 0E5H
ISP_TRIG        EQU 0E6H
ISP_CONTR       EQU 0E7H
;-------------------------------------------------
;定义常量
;------------------------
;Flash 操作等待时间
;ENABLE_ISP      EQU 83H             ;<5MHz
;ENABLE_ISP      EQU 82H             ;<10MHz
;ENABLE_ISP      EQU 81H             ;<20MHz
 ENABLE_ISP      EQU 80H             ;>20MHz
 DEBUG_DATA      EQU 5AH
;------------------------
;选择 MCU 型号
DATA_FLASH_START_ADDRESS EQU 2000H ;STC89C/LE52RC
;-------------------------------------------------
    ORG    0000H
    AJMP   MAIN
;-------------------------------------------------
    ORG    0100H
MAIN:
    MOV      P1,#0F0H ;演示程序开始工作
    LCALL    Delay    ;延时
    MOV      P1,#0FH  ;演示程序开始工作
    LCALL    Delay    ;延时
    MOV      SP,#0E0H ;堆栈指针指向 0E0H 单元
;*************************************************
;读回写入 flash 的第1个字节
MAIN1:
    MOV    DPTR, #DATA_FLASH_START_ADDRESS
    LCALL Byte_Read
    MOV    40H, A                ;值送40H 单元保存
    CJNE   A, #DEBUG_DATA, DATA_NOT_EQU_DEBUG_DATA
```

```
DATA_IS_DEBUG_DATA:
    MOV      P1,  #01111111B  ; (DATA_FLASH_START_ADDRESS) = #5A, 亮P1.7
    LCALL    Delay     ;延时
    MOV      A, 40H   ;值从40H 单元送 ACC
    CPL      A
    MOV      P1,A     ;数据是对的，送 P1显示
WAIT1:
    SJMP     WAIT1    ;数据是对的，送 P1显示，并在此停止

DATA_NOT_EQU_DEBUG_DATA:
    MOV      P1,  #11110111B ;(DATA_FLASH_START_ADDRESS) != #5A, 亮P1.3
    LCALL    Delay     ;延时

    MOV      A,   40H ;值从40H 单元送 ACC
    CPL      A
    MOV      P1,  A  ;数据不对，送 P1显示
    LCALL    Delay     ;延时

    MOV      DPTR,#DATA_FLASH_START_ADDRESS
    ACALL    Sector_Erase   ;擦除扇区, (DATA_FLASH_START_ADDRESS) != #DEBUG_DATA

    MOV      DPTR, #DATA_FLASH_START_ADDRESS
    MOV      A,    #DEBUG_DATA             ;写入 flash 的数据为 DEBUG_DATA
    ACALL    Byte_Program                  ;字节编程
    MOV      P1,   #11011111B              ;先亮P1.3 ,再亮 P1.5
WAIT2:
    SJMP     WAIT2        ;字节编程后在此停止

;*************************************************
;-----------------------------------------------
;读一字节
;调用前需打开 IAP 功能
;入口:DPTR = 字节地址
;返回:A = 读出字节

Byte_Read:
    MOV      ISP_CONTR,  #ENABLE_ISP      ;打开 IAP 功能，设置 Flash 操作等待时间
    MOV      ISP_CMD,    #01              ;Select Read AP Mode
    MOV      ISP_ADDRH,  DPH              ;Fill page address in ISP_ADDRH & ISP_ADDRL
    MOV      ISP_ADDRL,  DPL
    CLR      EA
    MOV      ISP_TRIG,   #46H             ;Trigger ISP processing
    MOV      ISP_TRIG,   #0B9H            ;Trigger ISP processing
    NOP
    MOV      A,   ISP_DATA                ;数据在ISP_DATA
    SETB     EA
;Now in processing.(CPU will halt here before completing)
```

```
    ACALL IAP_Disable                ;关闭 IAP 功能，清与 ISP 有关的特殊功能寄存器
    RET

;-----------------------------------------------
;字节编程
;调用前需打开 IAP 功能
;入口:DPTR = 字节地址，A=须编程字节的数据

Byte_Program:

    MOV    ISP_CONTR, #ENABLE_ISP    ;打开 IAP 功能，设置 Flash 操作等待时间
    MOV    ISP_CMD, #02H             ;Select Byte Program Mode
    MOV    ISP_ADDRH, DPH            ;Fill page address in ISP_ADDRH & ISP_ADDRL
    MOV    ISP_ADDRL, DPL
    MOV    ISP_DATA, A               ;数据进 ISP_DATA
    CLR    EA
    MOV    ISP_TRIG, #46H            ;Trigger ISP processing
    MOV    ISP_TRIG, #0B9H           ;Trigger ISP processing
    NOP
    SETB   EA
    ACALL IAP_Disable                ;关闭 IAP 功能，清与 ISP 有关的特殊功能寄存器
    RET
;-----------------------------------------------

;擦除扇区，入口:DPTR = 扇区地址
Sector_Erase:

    MOV    ISP_CONTR, #ENABLE_ISP    ;打开 IAP 功能，设置 Flash 操作等待时间
    MOV    ISP_CMD, #03H             ;Select Page Erase Mode
    MOV    ISP_ADDRH, DPH            ;Fill page address in ISP_ADDRH & ISP_ADDRL
    MOV    ISP_ADDRL, DPL
    CLR    EA
    MOV    ISP_TRIG, #46H            ;Trigger ISP processing
    MOV    ISP_TRIG, #0B9H           ;Trigger ISP processing
    NOP
    SETB   EA
    ACALL IAP_Disable                ;关闭 IAP 功能，清与 ISP 有关的特殊功能寄存器
    RET
;-----------------------------------------------
Trigger_ISP:
    CLR   EA
    MOV   ISP_TRIG, #46H             ;Trigger ISP processing
    MOV   ISP_TRIG, #0B9H            ;Trigger ISP processing
    NOP
    SETB  EA
    RET
```

```
;-----------------------------------------------
IAP_Disable:                        ;关闭 IAP 功能，清与 ISP 有关的特殊功能寄存器
    MOV   ISP_CONTR, #0             ;关闭 IAP 功能
    MOV   ISP_CMD, #0
    MOV   ISP_TRIG, #0
    RET

;-----------------------------------------------

Delay:
    CLR   A
    MOV   R0, A
    MOV   R1, A
    MOV   R2, #20H
Delay_Loop:
    DJNZ  R0, Delay_Loop
    DJNZ  R1, Delay_Loop
    DJNZ  R2, Delay_Loop
    RET

;-----------------------------------------------

    END
;*************************************************
```

STC89C51RC/RD+ 系列单片机定时器的使用

定时器 0 和 1

定时和计数功能由特殊功能寄存器 TMOD 的控制位 $C/\overline{T}$ 进行选择，TMOD 寄存器的各位信息如下表所列。可以看出，2 个定时 / 计数器有 4 种操作模式，通过 TMOD 的 M1 和 M0 选择。2 个定时 / 计数器的模式 0、1 和 2 都相同，模式 3 不同，各模式下的功能如下所述。

寄存器 TMOD 各位的功能描述

TMOD　　地址：89H　　复位值：00H

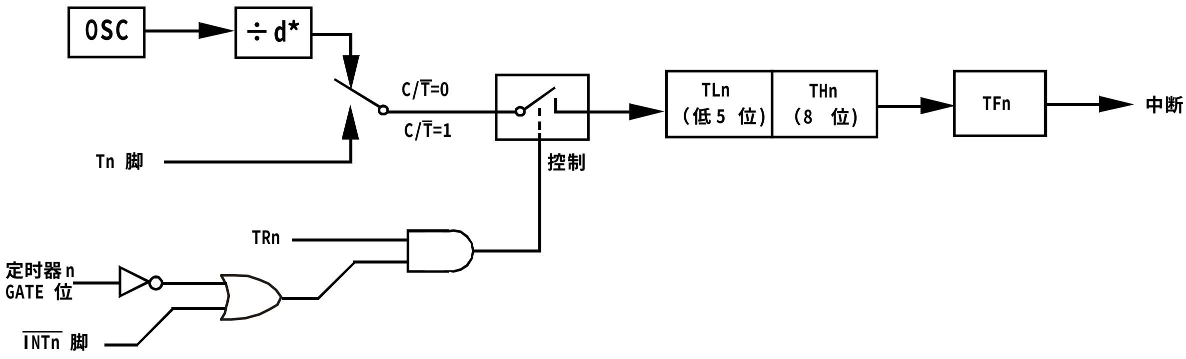
不可位寻址

7	6	5	4	3	2	1	0
GATE	$C/\overline{T}$	M1	M0	GATE	$C/\overline{T}$	M1	M0
定时器 1				定时器 0			

位	符号	功能
TMOD.7/	GATE	TMOD.7 控制定时器 1,置 1 时只有在 $\overline{INT1}$ 脚为高及 TR1 控制位置 1 时才可打开定时器 / 计数器 1。
TMOD.3/	GATE	TMOD.3 控制定时器 0,置 1 时只有在 $\overline{INT0}$ 脚为高及 TR0 控制位置 1 时才可打开定时器 / 计数器 0。
TMOD.6/	$C/\overline{T}$	TMOD.6 控制定时器 1 用作定时器或计数器，清零则用作定时器（从内部系统时钟输入），置 1 用作计数器（从 T1/P3.5 脚输入）
TMOD.2/	$C/\overline{T}$	TMOD.2 控制定时器 0 用作定时器或计数器，清零则用作定时器（从内部系统时钟输入），置 1 用作计数器（从 T0/P3.4 脚输入）
TMOD.5/TMOD.4	M1、M0	定时器 / 计数器 1 模式选择
	0　0	13 位定时器 / 计数器，兼容 8048 定时器模式，TL1 只用低 5 位参与分频，TH1 整个 8 位全用。
	0　1	16 位定时器 / 计数器，TL1、TH1 全用
	1　0	8 位自动重装载定时器，当溢出时将 TH1 存放的值自动重装入 TL1。
	1　1	定时器 / 计数器 1 此时无效（停止计数）。
TMOD.1/TMOD.0	M1、M0	定时器 / 计数器 0 模式选择
	0　0	13 位定时器 / 计数器，兼容 8048 定时器模式，TL0 只用低 5 位参与分频，TH0 整个 8 位全用。
	0　1	16 位定时器 / 计数器，TL0、TH0 全用
	1　0	8 位自动重装载定时器，当溢出时将 TH0 存放的值自动重装入 TL0。
	1　1	定时器 0 此时作为双 8 位定时器 / 计数器。TL0 作为一个 8 位定时器 / 计数器，通过标准定时器 0 的控制位控制。TH0 仅作为一个 8 位定时器，由定时器 1 的控制位控制。

1. 模式 0

将定时器设置成模式 0 时类似 8048 定时器，即 8 位计数器带 32 分频的预分频器。下图所示为模式 0 工作方式。此模式下，定时器配置为 13 位的计数器，由 TLn 的低 5 位和 THn 的 8 位所构成。TLn 低 5 位溢出向 THn 进位，THn 计数溢出置位 TCON 中的溢出标志位 TFn（n=0，1)。GATE=0 时，如 TRn=1，则定时器计数。GATE=1 时，允许由外部输入 $\overline{INT1}$ 控制定时器 1，$\overline{INT0}$ 控制定时器 0，这样可实现脉宽测量。TRn 为 TCON 寄存器内的控制位，TCON 寄存器各位的具体功能描述见 TCON 寄存器各位的具体功能描述表。



* 在6时钟模式下，d=6；在12时钟模式下，d=12。

图 定时器/计数器0和定时器/计数器1的模式0 ： 13位定时/计数器

表 寄存器TCON各位的功能描述

TCON 地址：88H

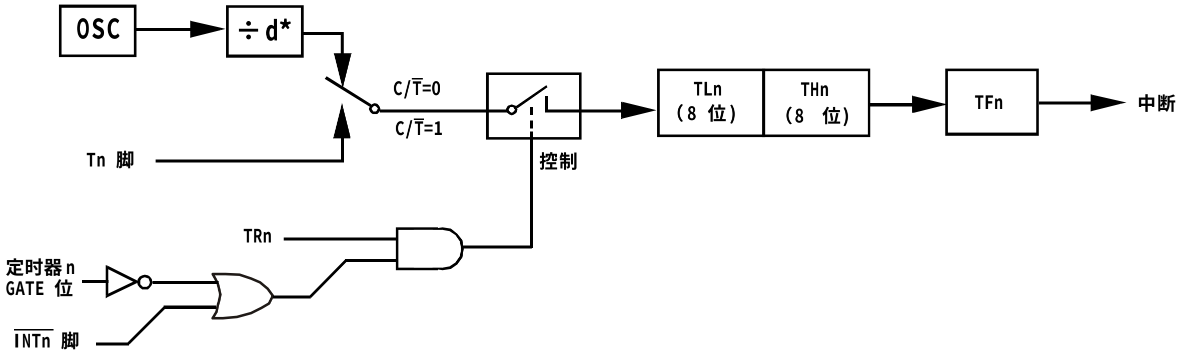
可位寻址 复位值：00H	7	6	5	4	3	2	1	0
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

位	符 号	功 能
TCON.7	TF1	定时器/计数器1溢出标志位。当T1被允许计数后，T1从初值开始加1计数，最高位产生溢出时，置“1”TF1，并向CPU请求中断，当CPU响应时，由硬件清“0”TF1，TF1也可以由程序查询或清“0”。
TCON.6	TR1	定时器T1的运行控制位。该位由软件置位和清零。当GATE（TMOD.7）=0，TR1=1时就允许T1开始计数，TR1=0时禁止T1计数。当GATE（TMOD.7）=1，TR1=1且INT1输入高电平时，才允许T1计数。
TCON.5	TF0	定时器/计数器0溢出标志位。当T0被允许计数后，T0从初值开始加1计数，最高位产生溢出时，置“1”TF0，并向CPU请求中断，当CPU响应时，由硬件清“0”TF0，TF0也可以由程序查询或清“0”。
TCON.4	TR0	定时器T0的运行控制位。该位由软件置位和清零。当GATE（TMOD.3）=0，TR0=1时就允许T0开始计数，TR1=0时禁止T0计数。当GATE（TMOD.3）=1，TR0=1且$\overline{INT0}$输入高电平时，才允许T0计数。
TCON.3	IE1	外部中断1中断请求标志位。当主机响应中断转向该中断服务程序执行时，由内部硬件自动将IE1位清0。
TCON.2	IT1	外部中断1触发方式控制位。IT1=0时，外部中断1为低电平触发方式，当$\overline{INT1}$（P3.3）输入低电平时，置位IE1。采用低电平触发方式时，外部中断源（输入到$\overline{INT1}$）必须保持低电平有效，直到该中断被CPU响应，同时在该中断服务程序执行完之前，外部中断源必须被清除（P3.3要变高），否则将产生另一次中断。当IT1=1时，则外部中断1（$\overline{INT1}$）端口由“1”→“0”下降沿跳变，激活中断请求标志位IE1，向主机请求中断处理。
TCON.1	IE0	外部中断0中断请求标志位。当主机响应中断转向该中断服务程序执行时，由内部硬件自动将IE0位清0。
TCON.0	IT0	外部中断1触发方式控制位。IT0=0时，外部中断0为低电平触发方式，当$\overline{INT0}$（P3.2）输入低电平时，置位IE0。采用低电平触发方式时，外部中断源（输入到$\overline{INT0}$）必须保持低电平有效，直到该中断被CPU响应，同时在该中断服务程序执行完之前，外部中断源必须被清除（P3.2要变高），否则将产生另一次中断。当IT0=1时，则外部中断0（$\overline{INT0}$）端口由“1”→“0”下降沿跳变，激活中断请求标志位IE1，向主机请求中断处理。

该13位寄存器包含THn全部8个位及TLn的低5位。TLn的高3位不定，可将其忽略。置位运行标志（TRn）不能清零此寄存器。模式0的操作对于定时器0及定时器1都是相同的。2个不同的GATE位（TMOD.7和TMOD.3）分别分配给定时器1及定时器0。

2．模式 1

模式 1 除了使用了 THn 及 TLn 全部 16 位外，其他与模式 0 完全相同。

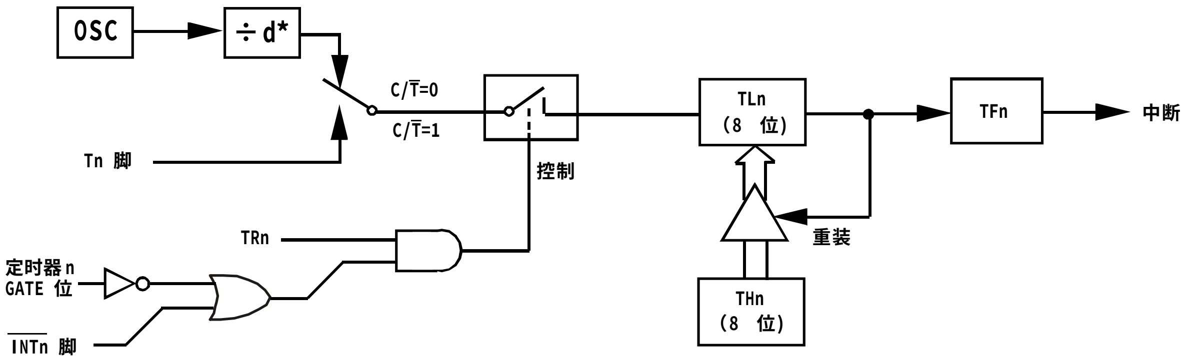


* 在 6 时钟模式下，d=6；在 12 时钟模式下，d=12。

图 定时器 / 计数器 0 和定时器 / 计数器 1 的模式 1 ： 16 位定时 / 计数器

3．模式 2

此模式下定时器 / 计数器 0 和 1 作为可自动重装载的 8 位计数器（TLn），如下图所示。TLn 的溢出不仅置位 TFn，而且将 THn 内容重新装入 TLn，THn 内容由软件预置，重装时 THn 内容不变。模式 2 的操作对于定时器 0 及定时器 1 是相同的。



* 在 6 时钟模式下，d=6；在 12 时钟模式下，d=12。

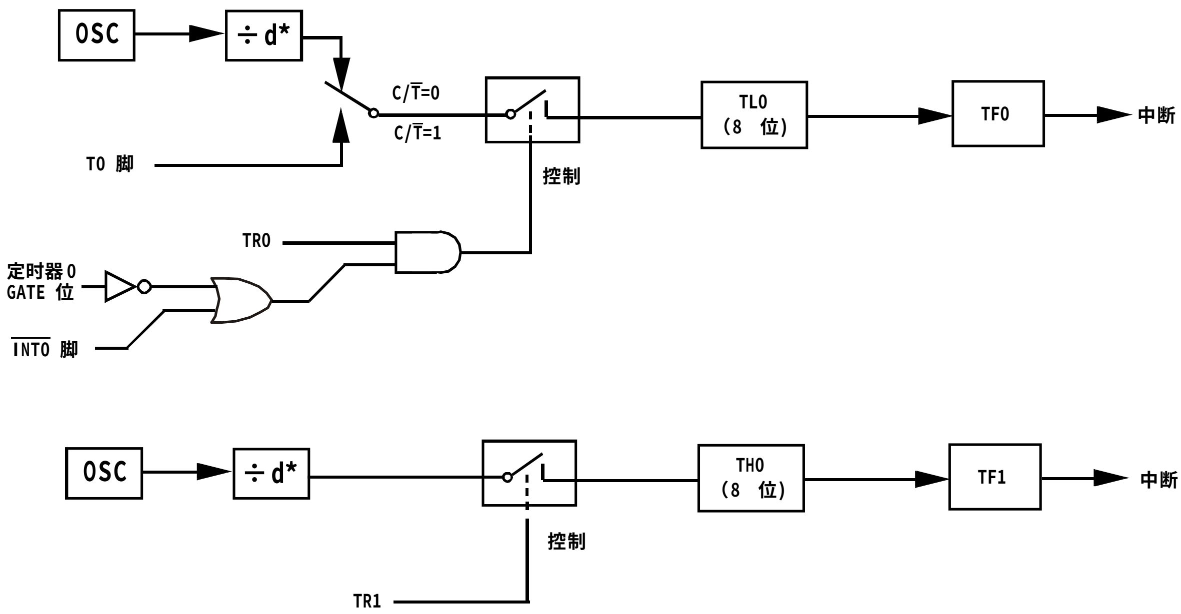
图 定时器 / 计数器 0 和 1 的模式 2 ： 8 位自动重装载

4.模式3

对定时器1，在模式3 时，定时器1 停止计数，效果与将TR1 设置为0 相同。

对定时器0，此模式下定时器0 的TL0 及TH0 作为2 个独立的8 位计数器。下图为模式3 时的定时器0 逻辑图。TL0 占用定时器0 的控制位：C/$\overline{T}$、GATE、TR0、$\overline{INT0}$ 及TF0。TH0 限定为定时器功能（计数器周期），占用定时器1 的TR1 及TF1。此时，TH0 控制定时器1 中断。

模式3 是为了增加一个附加的8 位定时器 / 计数器而提供的，使单片机具有三个定时器 / 计数器。模式3 只适用于定时器 / 计数器0，定时器T1 处于模式3 时相当于TR1=0，停止计数（此时T1 可用来作串行口波特率发生器），而T0 可作为两个定时器用。



* 在6 时钟模式下，d=6；在12 时钟模式下，d=12。

图 定时/计数器0 的模式3 ： 两个8 位计数器

定时器 0 和 1 应用举例

【例 1】 定时 / 计数器编程，定时 / 计数器的应用编程主要需考虑：根据应用要求，通过程序初始化，正确设置控制字，正确计算和计算计数初值，编写中断服务程序，适时设置控制位等。通常情况下，设置顺序大致如下：

1) 工作方式控制字（TMOD、T2CON）的设置；

2) 计数初值的计算并装入 THx、TLx、RCAP2H、RCAP2L；

3) 中断允许位 ETx、EA 的设置，使主机开放中断；

4) 启 / 停位 TRx 的设置等。

现以定时 / 计数器 0 或 1 为例作一简要介绍。

8051 系列单片机的定时器 / 计数器 0 或 1 是以不断加 1 进行计数的，即属加 1 计数器，因此，就不能直接将实际的计数值作为计数初值送入计数寄存器 THx、TLx 中去，而必须将实际计数值以 2^8、2^{13}、2^{16} 为模求补，以其补码作为计数初值设置 THx 和 TLx。

设：实际计数值为 X，计数器长度为 n（n=8、13、16），则应装入计数器 THx、TLx 中的计数初值为 2^n-x，式中 2^n 为取模值。例如，工作方式 0 的计数长度为 13 位，则 n=13，以 2^{13} 为模，工作方式 1 的计数长度为 16，则 n=16，以 2^{16} 为模等等。所以，计数初值为（x）= 2^n-x。

对于定时模式，是对机器周期计数，而机器周期与选定的主频密切相关。因此，需根据应用系统所选定的主频计算出机器周期值。现以主频 6MHz 为例，则机器周期为：

$$一个机器周期 = \frac{12}{主振频率} = \frac{12}{6 \times 10^6} \mu s = 2 \mu s$$

实际定时时间 Tc = x · Tp

式中 Tp 为机器周期，Tc 为所需定时时间，x 为所需计数次数。Tp 和 Tp 一般为已知值，在求出 Tp 后即可求得所需计数值 x，再将 x 求补码，即求得定时计数初值。即

$$(x)补 = 2^n - x$$

例如，设定时时间 Tc = 5ms，机器周期 TP = 2 μs，可求得定时计数次数

$$x = \frac{5ms}{2 \mu s} = 2500 次$$

设选用工作方式 1，则 n=16，则应设置的定时时间计数初值为：(x) 补 = 2^{16} - x=65536-2500=63036，还需将它分解成两个 8 位十六进制数，分别求得低 8 位为 3CH 装入 TLx，高 8 位为 F6H 装入 THx 中。

工作方式 0、1、2 的最大计数次数分别为 8192、65536 和 256。

对外部事件计数模式，只需根据实际计数次数求补后变换成两个十六进制码即可。

【例 2】 定时 / 计数器应用编程，设某应用系统，选择定时 / 计数器 1 定时模式，定时时间 Tc = 10ms，主频频率为 12MHz，每 10ms 向主机请求处理。选定工作方式 1。计算得计数初值：低 8 位初值为 F0H，高 8 位初值为 D8H。

（1） 初始化程序

所谓初始化，一般在主程序中根据应用要求对定时 / 计数器进行功能选择及参数设定等预置程序，本例初始化程序如下：

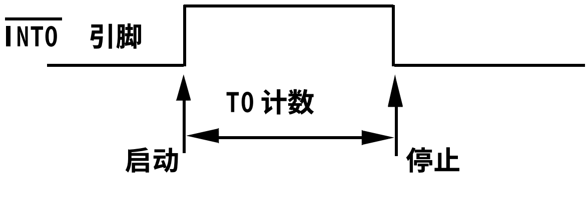
```
START：
          ⋮                     ；主程序段
          MOV   SP，#60H        ；设置堆栈区域
          MOV   TMOD，#10H      ；选择 T1 、定时模式，工作方式 1
          MOV   TH1，#0D8H      ；设置高字节计数初值
          MOV   TL1，#0F0H      ；设置低字节计数初值
          SETB  EA              } 开中断
          SETB  ET1             }
          ⋮                     ；其他初始化程序
          SETB  TR1             ；启动 T1 开始计时
          ⋮                     ；继续主程序
```

（2 ）中断服务程序

```
INTT1：   PUSH  A               }
          PUSH  DPL             } 现场保护
          PUSH  DPH             }
          ⋮
          MOV   TL1,#0F0H       } 重新置初值
          MOV   TH1,#0D8H       }
          ⋮                     ；中断处理主体程序
          POP   DPH             }
          POP   DPL             } 现场恢复
          POP   A               }
          RETI                  ；返回
```

这里展示了中断服务子程序的基本格式。8052 系列单片机的中断属于矢量中断，每一个矢量中断源只留有 8 个字节单元，一般是不够用的，常需用转移指令转到真正的中断服务子程序区去执行。

【例 3】　对外部正脉冲测宽。选择定时 / 计数器 2 进行脉宽测试较方便，但也可选用定时 / 计数器 0 或定时 / 计数器 1 进行测宽操作。本例选用定时 / 计数器 0（T0）以定时模式，工作方式 1 对 $\overline{INT0}$ 引脚上的正脉冲进行脉宽测试。



设置 GATE 为 1，机器周期 TP 为 1 μs。本例程序段编制如下：

```
INTT0：   MOV   TMOD，#09H      ；设 T0 为定时方式 1，GATE 为 1
```

```
            MOV   TL0，#00H          ;}  TH0,TL0 清 0
            MOV   TH0，#00H          ;}
            CLR   EX0                ;关 INT0 中断
LOP1：      JB    P3.2，LOP1         ;等待 INT0 引低电平
LOP2：      JNB   P3.2，LOP2         ;等待 INT0 引脚高电平
            SETB  TR0                ;启动 T0 开始计数
LOP3：      JB    P3.2，LOP3         ;等待 INT0 低电平
            CLR   TR0                ;停止 T0 计数
            MOV   A，TL0             ;低字节计数值送 A
            MOV   B，TH0             ;高字节计数值送 B
             ⋮                       ;计算脉宽和处理
```

【例 4】 利用定时 / 计数器 0 或定时 / 计数器 1 的 Tx 端口改造成外部中断源输入端口的应用设计。

在某些应用系统中常会出现原有的两个外部中断源 INT0 和 INT1 不够用，而定时 / 计数器有多余，则可将 Tx 用于增加的外部中断源。现选择定时 / 计数器 1 为对外部事件计数模式工作方式 2（自动再装入），设置计数初值为 FFH，则 T1 端口输入一个负跳变脉冲，计数器即回 0 溢出，置位对应的中断请求标志位 TF1 为 1，向主机请求中断处理，从而达到了增加一个外部中断源的目的。应用定时 / 计数器 1（T1）的中断矢量转入中断服务程序处理。其程序示例如下：

（1）主程序段：

```
      ORG   0000H
      AJMP  MAIN             ;转主程序
      ORG   001BH
      LJMP  INTER            ;转 T1 中断服务程序
       ⋮
      ORG   0100             ;主程序入口
MAIN：…
       ⋮
      MOV   SP，#60H         ;设置堆栈区
      MOV   TMOD，#60H       ;设置定时 / 计数器 1，计数方式 2
      MOV   TL1，#0FFH       ;设置计数常数
      MOV   TH1，#0FFH
      SETB  EA               ;开中断
      SETB  ET1              ;开定时 / 计数器 1 中断
      SETB  TR1              ;启动定时 / 计数器 1 计数
       ⋮
```

（2）中断服务程序（具体处理程序略）

```
            ORG   1000H
INTER：     PUSH  A            ;}
            PUSH  DPL          ;}  现场入栈保护
            PUSH  DPH          ;}
             ⋮
```

```
        ⋮                        } 中断处理主体程序
        ⋮
    POP   DPH                    ;
    POP   DPL                    ;  } 现场出栈复原
    POP   A                      ;
    RETI                         ; 返回
```

这是中断服务程序的基本格式。

【例5】 某应用系统需通过P1.0和P1.1分别输出周期为200μs和400μs的方波。为此，系统选用定时器/计数器0（T0），定时方式3，主频为6MHz，TP=2μs，经计算得定时常数为9CH和38H。

本例程序段编制如下：

（1）初始化程序段

```
          ⋮
PLT0:MOV   TMOD,#03H          ;设置T0定时方式3
     MOV   TL0，#9CH          ;设置TL0初值
     MOV   TH0，#38H          ;设置TH0初值
     SETB  EA                 ;
     SETB  ET0                ;  } 开中断
     SETB  ET1                ;
     SETB  TR0                ;启动
     SETB  TR1                ;启动
          ⋮
```

(2)中断服务程序段

1)

```
INT0P：  ⋮
         ⋮
    MOV  TL0，#9CH            ;重新设置初值
    CPL  P1.0                 ;对P1.0输出信号取反
         ⋮
    RETI                      ;返回
```

2)

```
INT1P    ⋮
         ⋮
    MOV  TH0，#38H            ;重新设置初值
    CPL  P1.1                 ;对P1.1输出信号取反
         ⋮
    RETI                      ;返回
```

在实际应用中应注意的问题如下。

（1）定时 / 计数器的实时性

定时 / 计数器启动计数后，当计满回 0 溢出向主机请求中断处理，由内部硬件自动进行。但从回 0 溢出请求中断到主机响应中断并作出处理存在时间延迟，且这种延时随中断请求时的现场环境的不同而不同，一般需延时 3 个机器周期以上，这就给实时处理带来误差。大多数应用场合可忽略不计，但对某些要求实时性苛刻的场合，应采用补偿措施。

这种由中断响应引起的时间延时，对定时 / 计数器工作于方式 0 或 1 而言有两种含义：一是由于中断响应延时而引起的实时处理的误差；二是如需多次且连续不间断地定时 / 计数，由于中断响应延时，则在中断服务程序中再置计数初值时已延误了若干个计数值而引起误差，特别是用于定时就更明显。

例如选用定时方式 1 设置系统时钟，由于上述原因就会产生实时误差。这种场合应采用动态补偿办法以减少系统始终误差。所谓动态补偿，即在中断服务程序中对 THx、TLx 重新置计数初值时，应将 THx、TLx 从回 0 溢出又重新从 0 开始继续计数的值读出，并补偿到原计数初值中去进行重新设置。可考虑如下补偿方法：

```
        ⋮
CLR   EA              ；禁止中断
MOV   A，TLx          ；读 TLx 中已计数值
ADD   A，#LOW         ；LOW 为原低字节计数初值
MOV   TLx，A          ；设置低字节计数初值
MOV   A，#HIGH        ；原高字节计数初值送 A
ADDC  A，THx          ；高字节计数初值补偿
MOV   THx，A          ；置高字节计数初值
SETB  EA              ；开中断
        ⋮
```

（2）动态读取运行中的计数值

在动态读取运行中的定时 / 计数器的计数值时，如果不加注意，就可能出错。这是因为不可能在同一时刻同时读取 THx 和 TLx 中的计数值。比如，先读 TLx 后读 THx，因为定时 / 计数器处于运行状态，在读 TLx 时尚未产生向 THx 进位，而在读 THx 前已产生进位，这时读得的 THx 就不对了；同样，先读 THx 后读 TLx 也可能出错。

一种可避免读错的方法是：先读 THx，后读 TLx，将两次读得的 THx 进行比较；若两次读得的值相等，则可确定读的值是正确的，否则重复上述过程，重复读得的值一般不会再错。此法的软件编程如下：

```
RDTM： MOV  A，THx              ；读取 THx 存 A 中
       MOV  R0，TLx             ；读取 TLx 存 R0 中
       CJNE A，THx，RDTM        ；比较两次 THx 值，若相等，则读得的值正
                                ；确，程序往下执行，否则重读
       MOV  R1，A               ；将 THx 存于 R1 中
         ⋮
```

定时器 1 做波特率发生器

```
;/* --- STC International Limited ---------------------------------------- */
;/* --- 宏晶科技 姚永平 设计 2006/1/6 V1.0 ---------------------------- */
;/* --- 演示 STC89C/LE51RC/RD+ 系列 MCU 定时器 1 作波特率发生器功能 --------- */
;/* --- Mobile: 13922805190 -------------------------------------------- */
;/* --- Tel: 0755-82948409 Fax: 0755-82944243------------------------ */
;/* --- Web: www.mcu-memory.com ---------------------------------------- */
;/* --- 本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过 ---------- */
;/* --- 如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 --- */
;/* --- 如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序 --- */

;-----------------------------------------------------------------------------
;      本程序演示 STC12C2052AD、STC12C5410AD 系列单片机用定时器 1 作 RS-232 通信
;波特率发生器的使用方法，有关波特率自动重装数的计算请查看程序后面的内容
;      本程序同时演示 STC89C51RC/STC89C52RC/STC89C53RC/STC89C54RD+/STC89C58RD+/
;STC89C516RD+ 系列单片机用定时器 1 作 RS-232 通信波特率发生器的使用方法。
;      STC12C2052AD、STC12C5410AD 系列是 "一个时钟 / 机器周期" 的 8051 单片机。它
;的定时器 0、定时器 1 有两种计数速率，一种是 12T 模式：每 12 个时钟加 1，与普通的
;8051 单片机相同；另一种是 1T 模式：每个时钟加 1，是普通 8051 单片机的 12 倍。
;      STC89C51RC/RD+ 系列是 "12 个时钟 / 机器周期" 的 8051 单片机，与普通的 8051 单片
;机相同。
;      STC12C2052AD、STC12C5410AD 系列的单片机，定时器 0、定时器 1 的计数速率由
;特殊功能寄存器 AUXR 的 bit7，bit6 决定，bit7 的符号是 T0x12，如果 T0x12=1，
;定时器 0 就工作在 1T 模式。bit6 的符号是 T1x12，如果 T1x12=1，定时器 1 就工作在
;1T 模式。有关详情请参考 STC12C5410AD 系列单片机器件手册(中文应用指南)。

;使用方法:
; 1. 修改程序，改变波特率参数或改变定时器 1 的计数速率(1T 模式 /12T 模式)
; 2. 汇编程序，将代码下载到单片机中
; 3. 调整串口调试助手的波特率与单片机的波特率相同，并打开调试助手的串口。STC
;    下载程序 STC-ISP.exe 版本 3.2 以上有串口调试助手功能。
; 4. 打开单片机电源，可以在串口调试助手的接收区看到单片机发出的数据
; 5. 用串口调试助手发送单个字节到单片机，单片机收到后会立即回发到串口调试助手
; 6. 反复步骤 1-5，检验波特率参数是否正确，特别要观察定时器 1 工作在 1T 模式
;    的波特率。例如，先设置定时器 1 工作在 12T 模式，设置波特率为 9600，执行
;    步骤 2-5，检验波特率参数是否正确。然后仅仅将定时器 1 的计数速率改成
;    1T 模式，执行步骤 2-5，就会发现本程序的波特率变成了 115200，波特率是
;    12T 模式的 12 倍。
;
;-----------------------------------------------------------------------------
```

```
;定义 STC12C5410AD 系列 MCU 特殊功能寄存器
AUXR            EQU   8EH
;-----------------------------------------------------------------------------
;定义波特率自动重装数
;*****************************************************************************
;以下是 Fosc = 22.1184MHz, 1T 模式, SMOD=1 时, 计算出的自动重装数和波特率
;RELOAD_COUNT EQU 0FFH      ;Baud=1,382,400 bps
;RELOAD_COUNT EQU 0FEH      ;Baud=691,200 bps
;RELOAD_COUNT EQU 0FDH      ;Baud=460,800 bps
;RELOAD_COUNT EQU 0FCH      ;Baud=345,600 bps
;RELOAD_COUNT EQU 0FBH      ;Baud=276,480 bps
;RELOAD_COUNT EQU 0FAH      ;Baud=230,400 bps
;RELOAD_COUNT EQU 0F4H      ;Baud=115,200 bps
;RELOAD_COUNT EQU 0E8H      ;Baud=57,600 bps
;RELOAD_COUNT EQU 0DCH      ;Baud=38,400 bps
;RELOAD_COUNT EQU 0B8H      ;Baud=19,200 bps
;RELOAD_COUNT EQU 70H      ;Baud=9,600 bps
;以上是 Fosc = 22.1184MHz, 1T 模式, SMOD=1 时, 计算出的自动重装数和波特率
;*****************************************************************************

;*****************************************************************************
;以下是 Fosc = 1.8432MHz, 1T 模式, SMOD=1 时, 计算出的自动重装数和波特率
;RELOAD_COUNT EQU 0FFH      ;Baud=115,200 bps
;RELOAD_COUNT EQU 0FEH      ;Baud=57,600 bps
;RELOAD_COUNT EQU 0FDH      ;Baud=38,400 bps
;RELOAD_COUNT EQU 0FCH      ;Baud=28,800 bps
;RELOAD_COUNT EQU 0FAH      ;Baud=19,200 bps
;RELOAD_COUNT EQU 0F4H      ;Baud=9,600 bps
;RELOAD_COUNT EQU 0E8H      ;Baud=4,800 bps
;RELOAD_COUNT EQU 0D0H      ;Baud=2,400 bps
;RELOAD_COUNT EQU 0A0H      ;Baud=1,200 bps
;以上是 Fosc = 1.8432MHz, 1T 模式, SMOD=1 时, 计算出的自动重装数和波特率
;*****************************************************************************

;*****************************************************************************
;以下是 Fosc = 18.432MHz, 1T 模式, SMOD=1 时, 计算出的自动重装数和波特率
;RELOAD_COUNT EQU 0FFH      ;Baud=1,152,000 bps
;RELOAD_COUNT EQU 0FEH      ;Baud=576,000 bps
;RELOAD_COUNT EQU 0FDH      ;Baud=288,000 bps
;RELOAD_COUNT EQU 0FCH      ;Baud=144,000 bps
;RELOAD_COUNT EQU 0F6H      ;Baud=115,200 bps
;RELOAD_COUNT EQU 0ECH      ;Baud=57,600 bps
;RELOAD_COUNT EQU 0E2H      ;Baud=38,400 bps
;RELOAD_COUNT EQU 0D8H      ;Baud=28,800 bps
;RELOAD_COUNT EQU 0C4H      ;Baud=19,200 bps
;RELOAD_COUNT EQU 088H      ;Baud=9,600 bps
;以上是 Fosc = 18.432MHz, 1T 模式, SMOD=1 时, 计算出的自动重装数和波特率
;*****************************************************************************
```

```
;*****************************************************************************
;以下是 Fosc = 18.432MHz, 1T 模式, SMOD=0 时，计算出的自动重装数和波特率

;RELOAD_COUNT EQU 0FFH      ;Baud=576,000 bps
;RELOAD_COUNT EQU 0FEH      ;Baud=288,000 bps
;RELOAD_COUNT EQU 0FDH      ;Baud=144,000 bps
;RELOAD_COUNT EQU 0FCH      ;Baud=115,200 bps
;RELOAD_COUNT EQU 0F6H      ;Baud=57,600 bps
;RELOAD_COUNT EQU 0ECH      ;Baud=38,400 bps
;RELOAD_COUNT EQU 0E2H      ;Baud=28,800 bps
;RELOAD_COUNT EQU 0D8H      ;Baud=19,200 bps
;RELOAD_COUNT EQU 0C4H      ;Baud=9,600 bps
;RELOAD_COUNT EQU 088H      ;Baud=4,800 bps

;以上是 Fosc = 18.432MHz, 1T 模式, SMOD=0 时，计算出的自动重装数和波特率
;*****************************************************************************

;*****************************************************************************
;以下是 Fosc = 18.432MHz, 12T 模式, SMOD=0 时，计算出的自动重装数和波特率

RELOAD_COUNT EQU 0FBH       ;Baud=9,600 bps
;RELOAD_COUNT EQU 0F6H      ;Baud=4,800 bps
;RELOAD_COUNT EQU 0ECH      ;Baud=2,400 bps
;RELOAD_COUNT EQU 0D8H      ;Baud=1,200 bps

;以上是 Fosc = 18.432MHz, 12T 模式, SMOD=0 时，计算出的自动重装数和波特率
;*****************************************************************************

;*****************************************************************************
;以下是 Fosc = 18.432MHz, 12T 模式, SMOD=1 时，计算出的自动重装数和波特率

;RELOAD_COUNT EQU 0FBH      ;Baud=19,200 bps
;RELOAD_COUNT EQU 0F6H      ;Baud=9,600 bps
;RELOAD_COUNT EQU 0ECH      ;Baud=4,800 bps
;RELOAD_COUNT EQU 0D8H      ;Baud=2,400 bps
;RELOAD_COUNT EQU 0B0H      ;Baud=1,200 bps

;以上是 Fosc = 18.432MHz, 12T 模式, SMOD=1 时，计算出的自动重装数和波特率
;*****************************************************************************
```

```
;*******************************************************************************
;以下是 Fosc = 11.0592MHz, 12T 模式, SMOD=0 时，计算出的自动重装数和波特率

;RELOAD_COUNT EQU 0FFH       ;Baud=28,800 bps
;RELOAD_COUNT EQU 0FEH       ;Baud=14,400 bps
;RELOAD_COUNT EQU 0FDH       ;Baud=9,600 bps
;RELOAD_COUNT EQU 0FAH       ;Baud=4,800 bps
;RELOAD_COUNT EQU 0F4H       ;Baud=2,400 bps
;RELOAD_COUNT EQU 0E8H       ;Baud=1,200 bps

;以上是 Fosc = 11.0592MHz, 12T 模式, SMOD=0 时，计算出的自动重装数和波特率
;*******************************************************************************

;*******************************************************************************
;以下是 Fosc = 11.0592MHz, 12T 模式, SMOD=1 时，计算出的自动重装数和波特率

;RELOAD_COUNT EQU 0FFH       ;Baud=57,600 bps
;RELOAD_COUNT EQU 0FEH       ;Baud=28,800 bps
;RELOAD_COUNT EQU 0FDH       ;Baud=14,400 bps
;RELOAD_COUNT EQU 0FAH       ;Baud=9,600 bps
;RELOAD_COUNT EQU 0F4H       ;Baud=4,800 bps
;RELOAD_COUNT EQU 0E8H       ;Baud=2,400 bps
;RELOAD_COUNT EQU 0D0H       ;Baud=1,200 bps

;以上是 Fosc = 11.0592MHz, 12T 模式, SMOD=1 时，计算出的自动重装数和波特率
;*******************************************************************************

;定义指示灯
LED_MCU_START   EQU   P1.7              ;MCU 工作指示灯
;-------------------------------------------------------------------------------
    ORG    0000H
    AJMP   MAIN
;-------------------------------------------------------------------------------
    ORG    0023H
    AJMP   UART_Interrupt               ;RS232 串口中断服务程序
    NOP
    NOP
;-------------------------------------------------------------------------------
MAIN:
    MOV    SP, #7FH                     ;设置堆栈指针
    CLR    LED_MCU_START                ;点亮 MCU 工作指示灯
    ACALL  Initial_UART                 ;初始化串口
    MOV    R0, #30H                     ;30H = 可打印字符 '0' 的ASCII码
    MOV    R2, #10                      ;发送10个字符'0123456789'
```

```
LOOP:
    MOV    A, R0
    ACALL Send_One_Byte               ;发送一个字节,可将PC串口调试助手设置成字符显示
    ;如果是字符显示，显示为 0123456789,
    ;如设置成 16 进制显示，显示 30 31 32 33 34 35 36 37 38 39
    INC    R0
    DJNZ   R2, LOOP
MAIN_WAIT:
    SJMP   MAIN_WAIT                  ;跳转到本行，无限循环
;------------------------------------------------------------------------------
UART_Interrupt:                       ;串口中断服务程序
    JB     RI, Is_UART_Receive
    CLR    TI                         ;清零串口发送中断标志
    RETI                               ;发送时使用的是查询方式，不使用中断
Is_UART_Receive:
    CLR    RI
    PUSH   ACC
    MOV    A, SBUF                    ;取接收到的字节
    ACALL Send_One_Byte               ;回发收到的字节
    POP    ACC
    RETI
;------------------------------------------------------------------------------
Initial_UART:                         ;初始化串口
; SCON  Bit:   7       6      5      4      3      2      1     0
;            SM0/FE   SM1    SM2    REN    TB8    RB8    TI    RI
    MOV    SCON, #50H                 ; 0101,0000 8位可变波特率，无奇偶校验

    MOV    TMOD, #21H                 ;设置定时器1 为8位自动重装计数器
    MOV    TH1, #RELOAD_COUNT         ;设置定时器1 自动重装数
    MOV    TL1, #RELOAD_COUNT

;----------------------------------------
;     ORL    PCON, #80H               ;若本行有效，波特率可以加倍
;----------------------------------------
;以下两行指令只能有一行有效
;      ORL   AUXR, #01000000B          ;定时器1工作在1T模式，波特率可以快
     12倍
    ANL    AUXR, #10111111B            ;定时器1工作在12T模式，与普通的8051相同
;以上两行指令只能有一行有效
;----------------------------------------
    SETB   TR1                        ;启动定时器1
    SETB   ES
    SETB   EA
    RET
```

```
;-------------------------------------------------------------------------------
;入口参数: A = 要发送的字节
Send_One_Byte:                          ;发送一个字节
    CLR   ES
    CLR   TI                            ;清零串口发送中断标志
    MOV   SBUF, A
Wait_Send_Finish:
    JNB   TI, Wait_Send_Finish          ;等待发送完毕
    CLR   TI                            ;清零串口发送中断标志
    SETB  ES
    RET
;-------------------------------------------------------------------------------
    END
;-------------------------------------------------------------------------------
;计算自动重装数 RELOAD (SMOD = 0, SMOD 是 PCON 特殊功能寄存器的最高位):
;  1. 计算 RELOAD (以下是 SMOD = 0 时的计算公式)
;
;     a) 12T 模式的计算公式： RELOAD = 256 - INT(Fosc/Baud0/32/12 + 0.5)
;     b)  1T 模式的计算公式： RELOAD = 256 - INT(Fosc/Baud0/32 + 0.5)
;
;     式中: INT() 表示取整运算即舍去小数, 在式中加 0.5 可以达到四舍五入的目的
;           Fosc = 晶振频率
;           Baud0 = 标准波特率
;
;  2. 计算用 RELOAD 产生的波特率:
;       a)   Baud = Fosc/(256 - RELOAD)/32/12       12T 模式
;       b)   Baud = Fosc/(256 - RELOAD)/32           1T 模式
;
;  3. 计算误差
;          error = (Baud - Baud0)/Baud0 * 100%
;  4. 如果误差绝对值 > 4.5% 要更换波特率或者更换晶体频率, 重复步骤 1-4
;
;
;例: Fosc = 22.1184MHz, Baud0 = 57600 (12T 模式)
; 1. RELOAD = 256 - INT( 22118400/57600/32/12 + 0.5)
;           = 256 - INT( 1.5 )
;           = 256 - 1
;           = 255
;           = 0FFH
; 2. Baud = 22118400/(256-255)/32/12
;         = 57600
; 3. 误差等于零
```

```
;例: Fosc = 18.432MHz, Baud0 = 57600 (12T 模式)
; 1. RELOAD = 256 - INT( 18432000/57600/32/12 + 0.5)
;            = 256 - INT( 0.833 + 0.5 )
;            = 256 - INT( 1.333 )
;            = 256 - 1
;            = 255
;            = 0FFH
; 2. Baud = 18432000/(256-255)/32/12
;         = 48000
; 3. error = (48000 - 57600)/57600 * 100%
;          = -16.66%
; 4. 误差很大，要更换波特率或者更换晶体频率，重新计算请见下一例

;例: Fosc = 18.432MHz, Baud0 = 9600 (12T 模式)
; 1. RELOAD = 256 - INT( 18432000/9600/32/12 + 0.5)
;            = 256 - INT( 5.5 )
;            = 256 - 5
;            = 251
;            = 0FBH
; 2. Baud = 18432000/(256-251)/32/12
;         = 9600
; 3. 一目了然，误差等于零

;例: Fosc = 2.000MHz, Baud = 4800 (1T 模式)
; 1. RELOAD = 256 - INT( 2000000/4800/32 + 0.5)
;            = 256 - INT( 13.02 + 0.5 )
;            = 256 - INT( 13.52 )
;            = 256 - 13
;            = 243
;            = 0F3H
; 2. Baud = 2000000/(256-243)/32
;         = 4808
; 3. error = 0.16%
;---------------------------------------------------------------------------
```

STC 定时器 2 的操作

定时器 2 是一个 16 位定时 / 计数器。通过设置特殊功能寄存器 T2CON 中的 C/T2 位，可将其作为定时器或计数器（特殊功能寄存器 T2CON 的描述如表 1 所列）。定时器 2 有 3 种操作模式：捕获、自动重新装载（递增或递减计数）和波特率发生器，这 3 种模式由 T2CON 中的位进行选择（如表 1 所列）。

表 1　特殊功能寄存器 T2CON 的描述

T2CON　地址 =0C8H　可位寻址　　复位值 =00H

7	6	5	4	3	2	1	0
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$

符 号	位	名称和意义
TF2	T2CON.7	定时器 2 溢出标志。定时器 2 溢出时置位，必须由软件清除。当 RCLK 或 TCLK=1 时，TF2 将不会置位
EXF2	T2CON.6	定时器 2 外部标志。当 EXEN2=1 且 T2EX 的负跳变产生捕获或重装时，EXF2 置位。定时器 2 中断使能时，EXF2=1 将使 CPU 从中断向量处执行定时器 2 中断子程序。EXF2 位必须用软件清零。在递增 / 递减计数器模式（DCEN=1）中，EXF2 不会引起中断
RCLK	T2CON.5	接收时钟标志。RCLK 置位时，定时器 2 的溢出脉冲作为串行口模式 1 和模式 3 的接收时钟。RCLK=0 时，将定时器 1 的溢出脉冲作为接收时钟
TCLK	T2CON.4	发送时钟标志。TCLK 置位时，定时器 2 的溢出脉冲作为串行口模式 1 和模式 3 的发送时钟。TCLK=0 时，将定时器 1 的溢出脉冲作为发送时钟
EXEN2	T2CON.3	定时器 2 外部使能标志。当其置位且定时器 2 未作为串行口时钟时，允许 T2EX 的负跳变产生捕获或重装。EXEN2=0 时，T2EX 的跳变对定时器 2 无效
TR2	T2CON.2	定时器 2 启动 / 停止控制位。置 1 时启动定时器
C/$\overline{T2}$	T2CON.1	定时器 / 计数器选择。（定时器 2） 0= 内部定时器（OSC/12 或 OSC/6） 1 = 外部事件计数器（下降沿触发）
CP/$\overline{RL2}$	T2CON.0	捕获 / 重装标志。置位：EXEN2=1 时,T2EX 的负跳变产生捕获。清零：EXEN2=0 时，定时器 2 溢出或 T2EX 的负跳变都可使定时器自动重装。当 RCLK=1 或 TCLK=1 时，该位无效且定时器强制为溢出时自动重装

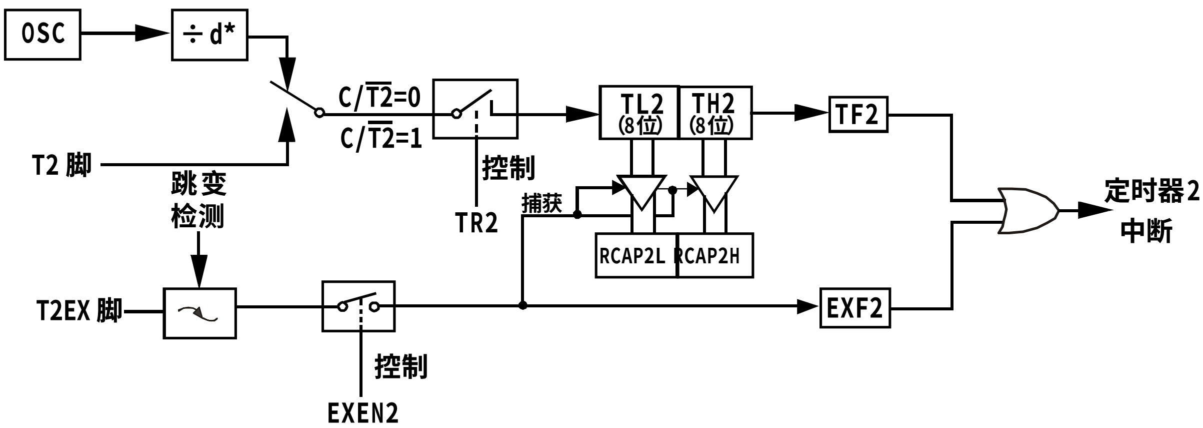
表 2　定时器 2 工作方式

RCLK+TCLK	CP/$\overline{RL2}$	TR2	模　式
0	0	1	16 位自动重装
0	1	1	16 位捕获
1	X	1	波特率发生器
X	X	0	（关闭）

1.捕获模式

在捕获模式中，通过 T2CON 中的 EXEN2 设置 2 个选项。如果 EXEN2=0，定时器 2 作为一个 16 位定时器或计数器（由 T2CON 中 C/$\overline{T2}$ 位选择），溢出时置位 TF2（定时器 2 溢出标志位）。该位可用于产生中断（通过使能 IE 寄存器中的定时器 2 中断使能位）。如果 EXEN2=1，与以上描述相同，但增加了一个特性，即外部输入 T2EX 由 1 变零时，将定时器 2 中 TL2 和 TH2 的当前值各自捕获到 RCAP2L 和

RCAP2H。另外，T2EX 的负跳变使 T2CON 中的 EXF2 置位，EXF2 也像 TF2 一样能够产生中断（其向量与定时器 2 溢出中断地址相同，定时器 2 中断服务程序通过查询 TF2 和 EXF2 来确定引起中断的事件），捕获模式如图 1 所示。在该模式中，TL2 和 TH2 无重新装载值，甚至当 T2EX 产生捕获事件时，计数器仍以 T2EX 的负跳变或振荡频率的 1/12（12 时钟模式）或 1/6（6 时钟模式）计数。



*　在 6 时钟模式下，d=6；在 12 时钟模式下，d=12。

图 1　　定时器 2 捕获模式

2. 自动重装模式（递增 / 递减计数器）

16 位自动重装模式中，定时器 2 可通过 C/T2 配置为定时器 / 计数器，编程控制递增 / 递减计数。计数的方向是由 DCEN（递减计数使能位）确定的，DCEN 位于 T2MOD 寄存器中，T2MOD 寄存器各位的功能描述如表 3 所示。当 DCEN=0 时，定时器 2 默认为向上计数；当 DCEN=1 时，定时器 2 可通过 T2EX 确定递增或递减计数。图 2 显示了当 DCEN=0 时，定时器 2 自动递增计数。在该模式中，通过设置 EXEN2 位进行选择。如果 EXEN2=0，定时器 2 递增计数到 0FFFFH，并在溢出后将 TF2 置位，然后将 RCAP2L 和 RCAP2H 中的 16 位值作为重新装载值装入定时器 2。RCAP2L 和 RCAP2H 的值是通过软件预设的。

表 3　　定时器 2 模式（T2MOD）控制寄存器的描述

T2MOD　地址 =0C9H　　　　复位值 =XXXX XX00B

不可位寻址

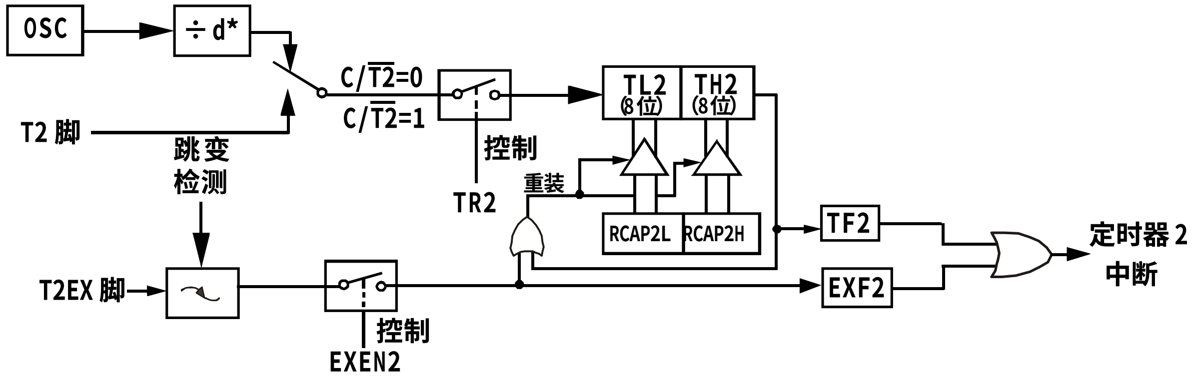
7	6	5	4	3	2	1	0
-	-	-	-	-	-	T2OE	DCEN

符 号	功 能
—	不可用，保留将来之用 *
T2OE	定时器 2 输出使能位
DCEN	向下计数使能位。定时器 2 可配置成向上 / 向下计数器

* 用户勿将其置 1。这些位在将来 80C51 系列产品中用来实现新的特性。在这种情况下，以后用到保留位，复位时或非有效状态时，它的值应为 0；而这些位为有效状态时，它的值为 1。从保留位读到的值是不确定的。

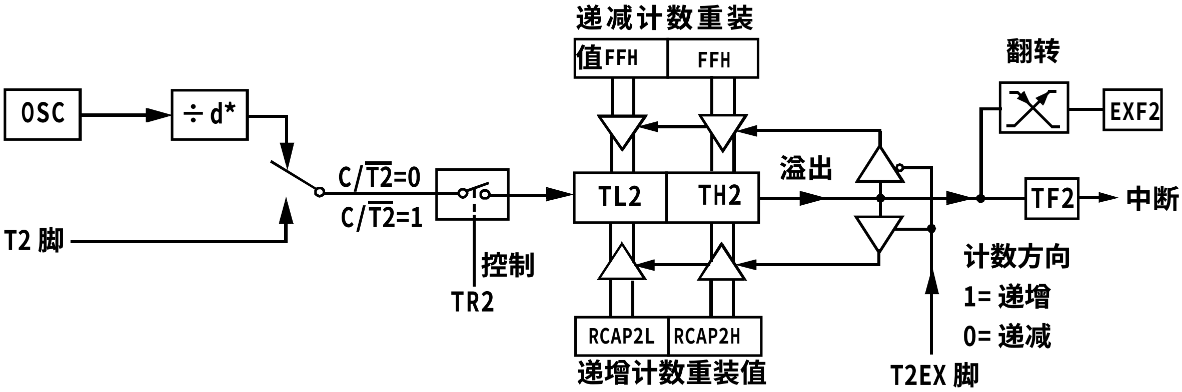
如果 EXEN2=1，16 位重新装载可通过溢出或 T2EX 从 1　0 的负跳变实现。此负跳变同时将 EXF2 置位。如果定时器 2 中断被使能，则当 TF2 或 EXF2 置 1 时产生中断。在图 3 中，DCEN=1 时，定时器 2 可递增或递减计数。此模式允许 T2EX 控制计数的方向。当 T2EX 置 1 时，定时器 2 递增计数，计数到 0FFFFH 后溢出并置位 TF2，还将产生中断（如果中断被使能）。定时器 2 的溢出将使 RCAP2L 和 RCAP2H 中的 16 位值作为重新装载值放入 TL2 和 TH2。

当 T2EX 置零时，将使定时器 2 递减计数。当 TL2 和 TH2 计数到等于 RCAP2L 和 RCAP2H 时，定时器产



* 在 6 时钟模式下，d=6；在 12 时钟模式下，d=12。

图 2　定时器 2 自动重装模式（DCEN=0）



* 在 6 时钟模式下，d=6；在 12 时钟模式下，d=12。

图 3　定时器 2 自动重装模式（DCEN=1）

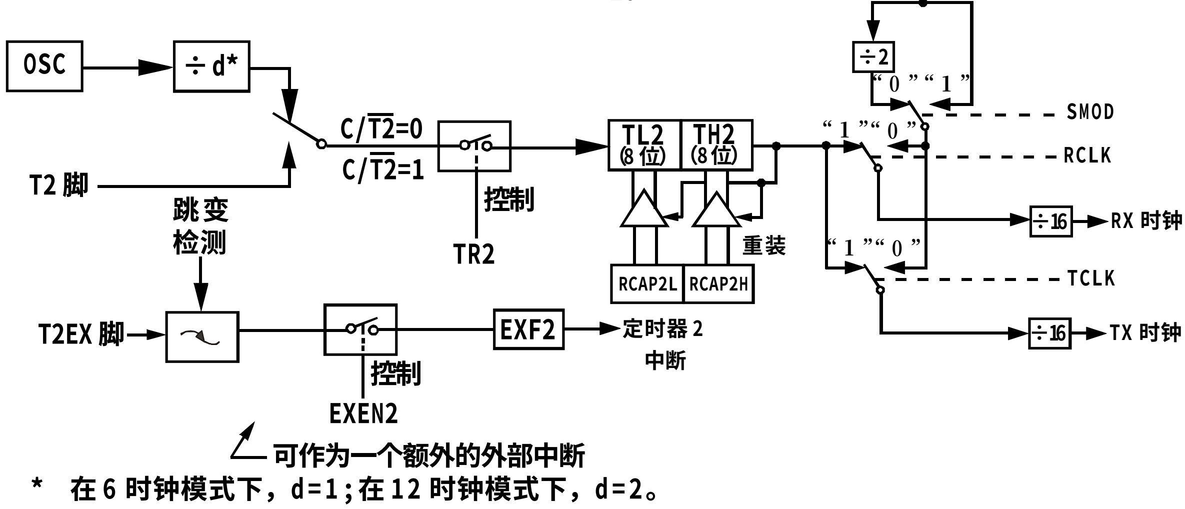
生中断。

3.波特率发生器模式

寄存器 T2CON 的位 TCLK 和（或）RCLK 允许从定时器 1 或定时器 2 获得串行口发送和接收的波特率。当 TCLK=0 时，定时器 1 作为串行口发送波特率发生器；当 TCLK=1 时，定时器 2 作为串行口发送波特率发生器。RCLK 对串行口接收波特率有同样的作用。通过这 2 位，串行口能得到不同的接收和发送波特率，一个通过定时器 1 产生，另一个通过定时器 2 产生。

如图 4 所示为定时器 2 工作在波特率发生器模式。与自动重装模式相似，当 TH2 溢出时，波特率发生器模式使定时器 2 寄存器重新装载来自寄存器 RCAP2H 和 RCAP2L 的 16 位的值,寄存器 RCAP2H 和 RCAP2L 的值由软件预置。当工作于模式 1 和模式 3 时，波特率由下面给出的公式所决定：

$$\text{模式 1 和模式 3 的波特率} = \frac{\text{定时器 2 溢出速率}}{16}$$



* 在 6 时钟模式下，d=1；在 12 时钟模式下，d=2。

图 4　定时器 2 波特率发生器模式

定时器可配置成“定时”或“计数”方式，在许多应用上，定时器被设置在“定时”方式（$C/\overline{T2}$=0）。当定时器 2 作为定时器时，它的操作不同于波特率发生器。通常定时器 2 作为定时器，它会在每个机器周期递增（1/6 或 1/12 振荡频率）。当定时器 2 作为波特率发生器时，它在 6 时钟模式下，以振荡器频率递增（12 时钟模式时为 1/12 振荡频率）。

这时的波特率公式如下：

$$\text{模式 1 和模式 3 的波特率} = \frac{\text{振荡器频率}}{n \times [65536 - (\text{RCAP2H}, \text{RCAP2L})]}$$

式中：n=16（6 时钟模式）或 32（12 时钟模式）；[RCAP2H，RCAP2L]是 RCAP2H 和 RCAP2L 的内容，为 16 位无符号整数。

如图 4 所示，定时器 2 是作为波特率发生器，仅当寄存器 T2CON 中的 RCLK 和（或）TCLK=1 时，定时器 2 作为波特率发生器才有效。注意：TH2 溢出并不置位 TF2，也不产生中断。这样当定时器 2 作为波特率发生器时，定时器 2 中断不必被禁止。如果 EXEN2（T2 外部使能标志）被置位，在 T2EX 中由 1 到 0 的转换会置位 EXF2（T2 外部标志位），但并不导致（TH2，TL2）重新装载（RCAP2H，RCAP2L）。当定时器 2 用作波特率发生器时，如果需要，T2EX 可用做附加的外部中断。

当计时器工作在波特率发生器模式下，则不要对 TH2 和 TL2 进行读 / 写，每隔一个状态时间（fosc/2)或由 T2 进入的异步信号，定时器 2 将加 1。在此情况下对 TH2 和 TH1 进行读 / 写是不准确的；可对 RCAP2 寄存器进行读，但不要进行写，否则将导致自动重装错误。当对定时器 2 或寄存器 RCAP 进行访问时，应关闭定时器（清零 TR2）。表 4 列出了常用的波特率和如何用定时器 2 得到这些波特率。

4.波特率公式汇总

定时器 2 工作在波特率发生器模式，外部时钟信号由 T2 脚进入，这时的波特率公式如下：

$$\text{波特率} = \frac{\text{定时器 2 溢出率}}{16}$$

如果定时器 2 采用内部时钟信号，则波特率公式如下：

$$\text{波特率} = \frac{fosc}{n \times [65536 - (\text{RCAP2H}, \text{RCAP2L})]}$$

表 4　由定时器 2 产生的常用波特率

波特率		振荡器频率 /MHz	定时器 2	
12 时钟模式	6 时钟模式		RCAP2H	RCAP2L
375 000	750 000	12	FF	FF
9 600	19 200	12	FF	D9
2 800	9 600	12	FF	B2
2 400	4 800	12	FF	64
1 200	2 400	12	FE	C8
300	600	12	FB	1E
110	220	12	F2	AF
300	600	6	FD	8F
110	220	6	F9	57

式中:n=32(12 时钟模式)或 16(6 时钟模式),fosc= 振荡器频率。

自动重装值可由下式得到:

$$\text{RCAP2H,RCAP2L} = 65536 - [fosc/(n \times \text{波特率})]$$

5.定时器 / 计数器 2 的设置

除了波特率发生器模式,T2CON 不包括 TR2 位的设置，TR2 位需单独设置来启动定时器。如表 5 和表 6 分别列出了 T2 作为定时器和计数器的具体设置方法。

表 5　T2 作为定时器的设置

模　　式	T2CON	
	内部控制①	外部控制②
16 位重装	00H	08H
16 位捕获	01H	09H
波特率发生器接收和发送相同波特率	34H	36H
只接收	24H	26H
只发送	14H	16H

① 仅当定时器溢出时进行捕获和重装。

② 当定时 / 计数器溢出并且 T2EX(P1.1)发生电平负跳变时产生捕获和重装(定时器 2 用于波特率发生器模式时除外)。

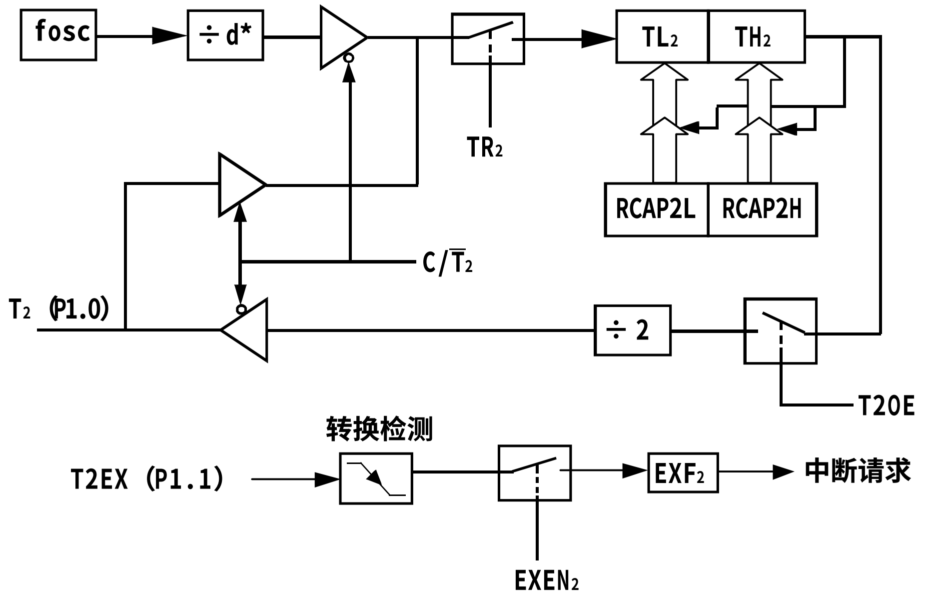
表 6　T2 作为计数器设置

模　　式	TMOD	
	内部控制①	外部控制②
16 位	02H	0AH
自动重装	03H	0BH

注①和注②同表 5 的注①和注②。

6.可编程时钟输出

STC89C51RC/RD+ 系列单片机，可设定定时 / 计数器 2，通过 P1.0 输出时钟。P1.0 除作通用 I/O 口外还有两个功能可供选用：用于定时 / 计数器 2 的外部计数输入和定时 / 计数器 2 时钟信号输出。图 5 为时钟输出和外部事件计数方式示意图。



*　d=1，6 时钟 / 机器周期；d=2，12 时钟 / 机器周期

图 5　定时器 2 时钟输出和外部事件计数方式示意图

通过软件对 T2CON.1 位 $C/\overline{T_2}$ 复位为 0，对 T2MOD.1 位 T2OE 置 1 就可将定时 / 计数器 2 选定为时钟信号发生器，而 T2CON.2 位 TR_2 控制时钟信号输出开始或结束（TR_2 为启 / 停控制位）. 由主振频率（fosc）和定时 / 计数器 2 定时、自动再装入方式的计数初值决定时钟信号的输出频率。其设置公式如下：

$$\text{时钟信号输出频率} = \frac{\text{fosc}}{n \times [65536 - (\text{RCAP2H}, \text{RCAP2L})]}$$

* n=2，6 时钟 / 机器周期；n=4，12 时钟 / 机器周期

从公式可见，在主振频率（fosc）设定后，时钟信号输出频率就取决于定时计数初值的设定。

在时钟输出模式下，计数器回 0 溢出不会产生中断请求。这种功能相当于定时 / 计数器 2 用作波特率发生器，同时又可以作时钟发生器。但必须注意，无论如何波特率发生器和时钟发生器不能单独确定各自不同的频率。原因是两者都用同一个陷阱寄存器 RCAP2H、RCAP2L，不可能出现两个计数初值。

```
;/* ---  STC International Limited ------------------------------------- */
;/* ---  宏晶科技 姚永平  设计 2006/1/6   V1.0 --------------------------- */
;/* ---  演示 STC89C/LE51RC/RD+ 系列 MCU 定时器2作波特率发生器功能--------- */
;/* ---  Mobile: 13922805190 ------------------------------------------- */
;/* ---  Tel: 0755-82948409    Fax: 0755-82944243------------------------ */
;/* ---  Web: www.mcu-memory.com ----------------------------------------- */
;/* ---  本演示程序在STC-ISP Ver 3.0A.PCB的下载编程工具上测试通过 ---------- */
;/* ---  如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 --- */
;/* ---  如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序 --- */

;-----Timer 2 做波特率发生器 -------------------------

;-----本程序不提供技术支持，一定要自己测试--------------

;定义特殊功能寄存器

;与 RS232口、TIMER2 有关的特殊功能寄存器
T2CON                     EQU 0C8H
TR2                       EQU T2CON.2   ;TR2 是 T2CON 特殊功能寄存器的第 2 位

RCAP2L                    EQU 0CAH
RCAP2H                    EQU 0CBH
TH2                       EQU 0CDH
TL2                       EQU 0CCH

;-----------------------------------------------
;设置波特率自动重装数
RELOAD_COUNT_HIGH         EQU  0FFH

;使用以下参数必须将 RELOAD_COUNT_HIGH 设置为 0FFH
;RELOAD_COUNT_LOW          EQU  0FAH    ;Fosc = 22.1184MHz, Baud = 115200
;RELOAD_COUNT_LOW          EQU  0EEH    ;Fosc = 22.1184MHz, Baud = 38400
;RELOAD_COUNT_LOW          EQU  0F0H    ;Fosc = 20.000MHz, Baud = 38400
;RELOAD_COUNT_LOW          EQU  0F6H    ;Fosc = 12.000MHz, Baud = 38400
;RELOAD_COUNT_LOW          EQU  0FDH    ;Fosc = 11.059MHz, Baud = 115200
;RELOAD_COUNT_LOW          EQU  0F7H    ;Fosc = 11.059MHz, Baud = 38400
;RELOAD_COUNT_LOW          EQU  0F8H    ;Fosc = 10.000MHz, Baud = 38400
;RELOAD_COUNT_LOW          EQU  0FBH    ;Fosc = 6.000MHz, Baud = 38400
;RELOAD_COUNT_LOW          EQU  0FCH    ;Fosc = 5.000MHz, Baud = 38400
;RELOAD_COUNT_LOW          EQU  070H    ;Fosc = 11.059MHz, Baud = 2400

;-------------
```

```
;计算自动重装数:
;--------------
;晶体频率: Fosc
;波特率:    Baud
;自动重装数:   RELOAD = INT(Fosc/Baud/32 + 0.5), INT 表示取整运算(舍去小数)
;将自动重装数转换成 16 进制, 用 10000H 减自动重装数, 存入 RCAP2H, RCAP2L
;计算实际的波特率: Baud = Fosc/RELOAD/32, 如果误差>3.5 要更改波特率.
;例: Fosc = 22.1184MHz, Baud = 115200
;  RELOAD = INT( 22118400/115200/32 + 0.5)
;         = INT( 6.5 )
;         = 6
;         = 0006H
; 10000H - 0006H = FFFAH
;
;   MOV   RCAP2H, #0FFH
;   MOV   RCAP2L, #0FAH
;例: Fosc = 20.MHz, Baud = 57600 (Baud=115200 时误差太大)
;  RELOAD = INT( 20000000/57600/32 + 0.5)
;         = INT( 10.85 + 0.5 )
;         = INT( 11.35 )
;         = 11
;         = 000BH
; 10000H - 000BH = FFF5H
;
;   MOV   RCAP2H, #0FFH
;   MOV   RCAP2L, #0F5H
;-------------------------------------------
    ORG   0000H
    AJMP  MAIN
;-------------------------------------------
    ORG   0023H                    ;RS232 串口中断
    AJMP  UART
    NOP
    NOP
;-------------------------------------------
MAIN:
    MOV   SP, #0E0H
    ACALL Initial_UART             ;初始化串口
    MOV   R0, #30H                 ;发送 10 个字符 ‘0123456789'
    MOV   R2, #10
LOOP:
    MOV   A, R0
    ACALL Send_One_Byte            ;发送一个字节
    INC   R0
    DJNZ  R2, LOOP
WAIT1：
    SJMP  WAIT1                    ;跳转到本行, 无限循环
```

```
;-------------------------------------------------
UART:                                   ;串口中断服务程序
    JBC    RI, UART_1
    RETI                                ;发送时使用的是查询方式，不使用中断
UART_1:                                 ;接收一个字节. 此时 RI 已被清0
    PUSH   ACC
    MOV    A, SBUF                      ;取接收到的字节
    ACALL  Send_One_Byte                ;回发收到的字节
    POP    ACC
    RETI
;-------------------------------------------------
Initial_UART:                           ;初始化串口
;        Bit:    7        6      5      4      3      2      1     0
; SCON          SM0/FE   SM1    SM2    REN    TB8    RB8    Tl    Rl
    MOV    SCON, #50H                   ; 0101,0000 8位可变波特率，无奇偶校验
Init_RS232:
    MOV    RCAP2H, A
    MOV    A, #RELOAD_COUNT_HIGH        ;波特率自动重装数
    MOV    TH2, A
    MOV    A, #RELOAD_COUNT_LOW
    MOV    RCAP2L, A
    MOV    TL2, A
    MOV    T2CON, #0x34                 ;使用 T2 作波特率发生器
    SETB   ES                           ;允许串口中断
    SETB   EA                           ;开总中断
    RET
;-------------------------------------------------
Send_One_Byte:                          ;发送一个字节
    CLR    ES
    CLR    TI                           ;清零串口发送中断标志
    MOV    SBUF, A

WAIT2：
    JNB    TI, WAIT2                    ;等待发送完毕
    CLR    TI                           ;清零串口发送中断标志
    SETB   ES
    RET
;-------------------------------------------------
    END
;-------------------------------------------------
```

定时器 2 的时钟输出功能,在 P1.0 口输出

```
;/* ---  STC International Limited ---------------------------------------- */
;/* ---  宏晶科技 姚永平   设计 2006/1/6    V1.0 ---------------------------- */
;/* ---  演示 STC89C51RC/RD+ 系列 MCU 定时器 2 的时钟输出功能,在 P1.0 口输出 --- */
;/* ---  Mobile: 13922805190 ---------------------------------------------- */
;/* ---  Tel: 0755-82948409     Fax: 0755-82944243------------------------ */
;/* ---  Web: www.mcu-memory.com ------------------------------------------ */
;/* ---  本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过 ---------- */
;/* ---  如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 --- */
;/* ---  如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序 --- */
;--------------------------------------------------------------------------------
;本程序演示了如何使用定时器 2 的时钟 / 脉冲输出功能，在 P1.0 口输出
;--------------------------------------------------------------------------------
;定义特殊功能寄存器
;与 RS232 口、TIMER2 有关的特殊功能寄存器
T2CON                    EQU 0C8H
T2MOD                    EQU 0C9H
TR2                      EQU T2CON.2    ;TR2 是 T2CON 特殊功能寄存器的第 2 位
RCAP2L                   EQU 0CAH
RCAP2H                   EQU 0CBH
TH2                      EQU 0CDH
TL2                      EQU 0CCH
;定时器 / 计数器 2 控制寄存器 T2CON
;                  D7     D6     D5     D4     D3     D2     D1     D0       Reset Value
;  位地址          CF     CE     CD     CC     CB     CA     C9     C8
; T2CON(C8H)       TF2    EXF2   RCLK   TCLK   EXEN2  TR2    C/T2   CP/RL2       00
;T2MOD 寄存器
;                  D7     D6     D5     D4     D3     D2     D1     D0       Reset Value
; T2CON(C9H)       -      -      -      -      -      -      T2OE   DCEN       xxxxxx00b
;---------------------------------------------------
    ORG    0000H
    AJMP   MAIN
;---------------------------------------------------
    ORG    0100H
MAIN:
    MOV    SP, #0E0H
    MOV    P1, #0FFH                ;熄灭 P1 口的发光二级管
    ACALL  SET_T2_OUT_MODE          ;设置 T2 为高速脉冲输出方式
    MOV    DPTR, #0FFF0H            ;设置 T2 脉冲输出速率
    ACALL  SET_T2_OUT_SPEED
WAIT1：
    SJMP   WAIT1                    ;可加入此行，用频率计或其它仪器测量 P1.0 的
                                    ;输出信号，验证脉冲输出频率的计算公式
    ACALL  DELAY
    ACALL  PAUSE                    ;暂停输出，便于观察
    MOV    DPTR, #0FFE0H            ;设置 T2 脉冲输出速率，比前一次降低一半
    ACALL  SET_T2_OUT_SPEED
```

```
    ACALL DELAY
    ACALL PAUSE                    ;暂停输出，便于观察
    MOV   DPTR, #0FFD0H            ;设置 T2 脉冲输出速率，比前一次降低 1/3
    ACALL SET_T2_OUT_SPEED
    ACALL DELAY
    ACALL PAUSE                    ;暂停输出，便于观察

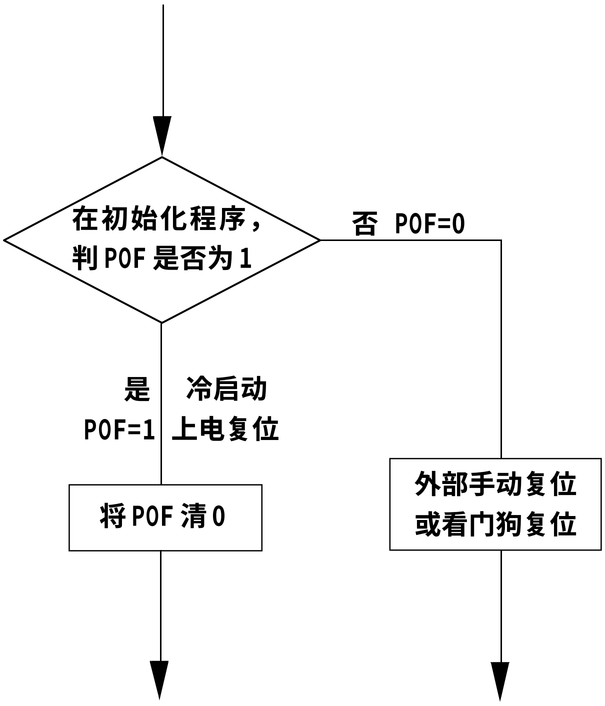
WAIT2：
    SJMP  WAIT2                    ;跳转到本行，无限循环
;-----------------------------------------------

DELAY:
    MOV   R1, #0
    MOV   R2, #0
    MOV   R3, #30
DELAY_LOOP:
    DJNZ  R1, DELAY_LOOP
    DJNZ  R2, DELAY_LOOP
    DJNZ  R3, DELAY_LOOP
    RET
;-----------------------------------------------
SET_T2_OUT_MODE:                   ;设置T2为脉冲输出方式
    MOV   T2CON, #0                ;设置T2为定时器方式
    MOV   T2MOD, #02               ;0000, 0010 允许 T2 溢出脉冲由 P1.0 输出
    RET
;-----------------------------------------------
;脉冲输出频率由振荡器频率和 T2 的捕获寄存器 RCAP2H、RCAP2L 的重新装入值确定，
;计算公式：
;     脉冲输出频率 = 振荡器频率 / (n*(65536 - RCAP2H,RCAP2L))
;公式中 n = 2, 在 6 Clock 模式; n = 4, 在 12 Clock 模式
;      RCAP2H,RCAP2L 是由 RCAP2H 和 RCAP2L 组成的 16 位无符号整数，
;入口: DPTR = 重装数
;对寄存器RCAP2H,RCAP2L不要送FFFFh，否则无脉冲输出
SET_T2_OUT_SPEED:                  ;设置 T2 脉冲输出速率
    CLR   TR2                      ;停止 T2 工作
    MOV   RCAP2H, DPH
    MOV   RCAP2L, DPL
    SETB  TR2                      ;启动 T2
    RET
;-----------------------------------------------
PAUSE:                             ;暂停输出
    CLR   TR2                      ;停止 T2 工作
    MOV   P1, #0FFH                ;熄灭 P1 口的发光二级管
    ACALL DELAY
    RET
;-----------------------------------------------
    END
;-----------------------------------------------
```

PCON 寄存器的高级应用，上电复位标志，进入掉电模式

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
PCON	87h	Power Control	SMOD	SMOD0	-	POF	GF1	GF0	PD	IDL	00x1,0000

POF：上电复位标志位，单片机停电后，上电复位标志位为 1，可由软件清 0。
实际应用：要判断是上电复位（冷启动），还是外部复位脚输入复位信号产生的复位，还是内部看门狗复位，可通过如下方法来判断：



PD：将其置 1 时，进入 Power Down 模式，可由外部中断低电平触发或下降沿触发中断模式唤醒。
进入掉电模式时，外部时钟停振，CPU、定时器、串行口全部停止工作，只有外部中断继续工作。
IDL：将其置 1 时，进入 IDLE 模式（空闲），除 CPU 不工作外，其余仍继续工作，可由任何一个中断唤醒。

STC89C51RC/RD+ 系列单片机如何从掉电模式唤醒

```
;************************************************************************
;Wake Up Idle and Wake Up Power Down
;************************************************************************
    ORG    0000H
    AJMP MAIN

    ORG    0003H
int0_interrupt:
    CLR    P1.7            ;点亮 P1.7 LED 表示已响应 int0 中断
    ACALL delay            ;延时是为了便于观察，实际应用不需延时
    CLR    EA              ;关闭中断，简化实验．实际应用不需关闭中断
    RETI

    ORG    0013H
int1_interrupt:
    CLR    P1.6            ;点亮 P1.6 LED 表示已响应 int1 中断
    ACALL delay            ;延时是为了便于观察，实际应用不需延时
    CLR    EA              ;关闭中断，简化实验．实际应用不需关闭中断
    RETI

    ORG    0100H
delay:
    CLR    A
    MOV    R0, A
    MOV    R1, A
    MOV    R2, #02
delay_loop:
    DJNZ   R0, delay_loop
    DJNZ   R1, delay_loop
    DJNZ   R2, delay_loop
    RET

main:
    MOV    R3, #0          ;P1 LED 递增方式变化，表示程序开始运行
main_loop:
    MOV    A, R3
    CPL    A
    MOV    P1, A
    ACALL delay
```

```
    INC     R3
    MOV     A, R3
    SUBB    A, #18H
    JC      main_loop

    MOV     P1, #0FFH         ;熄灭全部灯表示进入 Power Down 状态

    CLR     IT0               ;设置低电平激活外部中断
;   SETB    IT0
    SETB    EX0               ;允许外部中断 0

    CLR     IT1               ;设置低电平激活外部中断
;   SETB    IT1
    SETB    EX1               ;允许外部中断 1

    SETB    EA                ;开中断，若不开中断就不能唤醒 Power Down

;下条语句将使 MCU 进入 idle 状态或 Power Down 状态
;低电平激活外部中断可以将 MCU 从 Power Down 状态中唤醒
;其方法为:将外部中断脚拉低

    MOV     A, PCON           ;令 PD=1，进入 Power Down 状态，PD = PCON.1
    ORL     A, #02H
    MOV     PCON, A

    MOV     PCON, #01H        ;删除本语句前的";"，同时将前 3 条语句前加上注释符号";"，
                              ;令 IDL=1，可进入 idle 状态，IDL = PCON.0

    MOV     P1, #01010000B    ;请注意:
                              ; 1.外部中断使MCU退出 Power Down 状态,执行本条指令后
                              ;响应中断，表现为 P1.5 与 P1.7 的 LED 同时亮(INT0 唤醒)
                              ; 2.外部中断使MCU退出 idle 状态,先响应中断然后再执行本
                              ;条指令，表现为 P1.7 的 LED 先亮(INT0 唤醒)P1.5 的 LED 后亮

WAIT1：
    SJMP    WAIT1             ;跳转到本语句，停机

    END
```

STC89C51 RC/RD+系列单片机交直流特性

ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings

Parameter	Symbol	MIN	MAX	UNIT
Storage temperature	T_{ST}	-55	+125	℃
Operating Temperature(I)	T_A	-40	+85	℃
Operating Temperature(C)	T_A	0	+70	℃
DC Power Supply(5V MCU)	V_{DD} - V_{SS}	-0.3	+6.0	V
DC Power Supply(3V MCU)	V_{DD} - V_{SS}	-0.3	+4.0	V
Voltage on any Pin		-0.5	+5.5	V

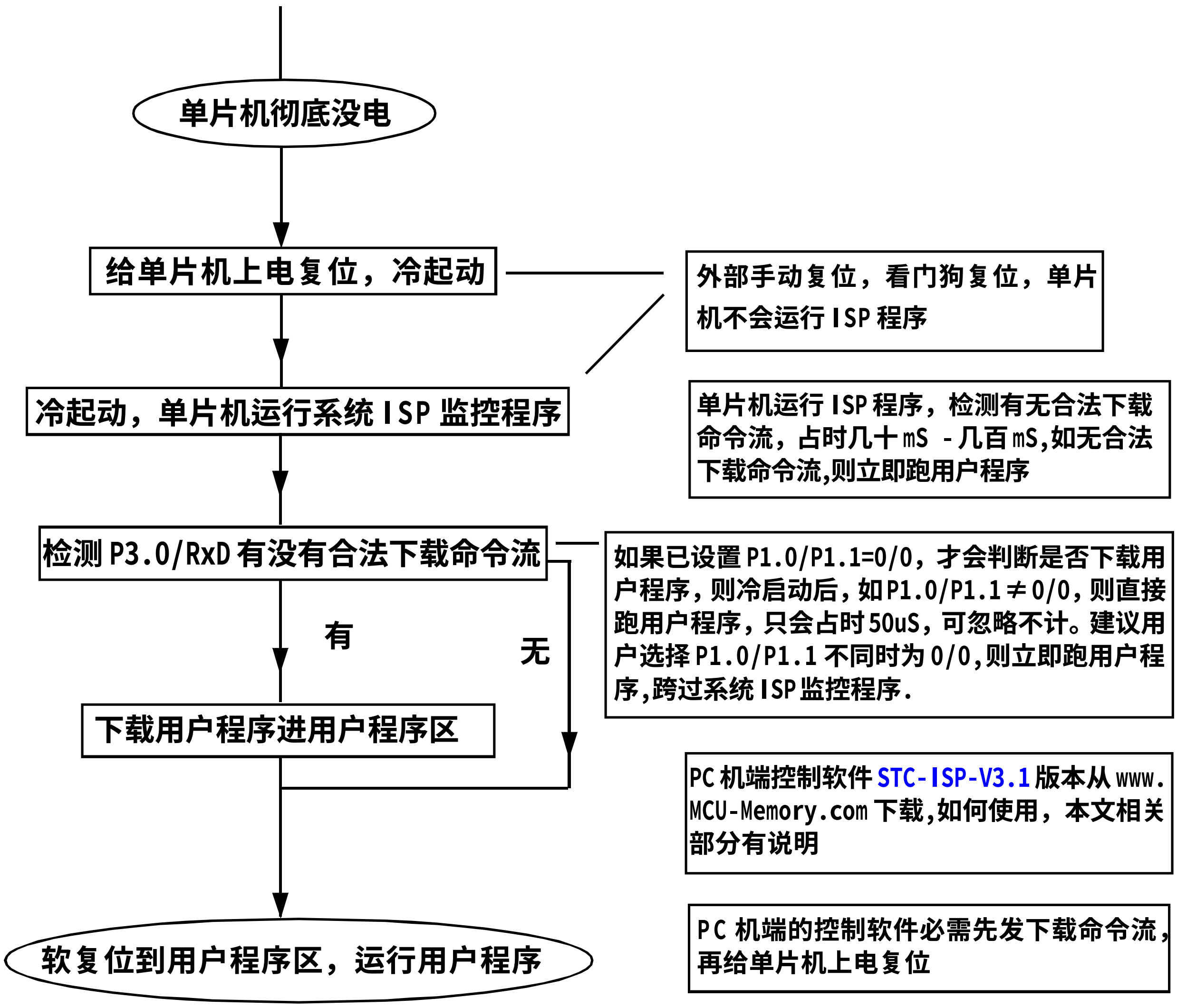
DC Specification(5V MCU)

Symbol	Parameter	Specification				Test Condition
		Min.	Typ.	Max.	Unit	
V_{DD}	Operating Voltage	3.8	5.0	5.5	V	
I_{PWDN}	Power Down Current		<0.1		uA	5V
I_{IDLE}	Idle Current		2.0		mA	5V
I_{CC}	Operating Current		4 mA	20	mA	5V
V_{IL1}	Input low voltage (P0,1,2,3,4)			0.8	V	5V
V_{IL2}	Input low voltage (RESET,XTAL1)			1.5	V	5V
V_{IH1}	Input High voltage (P0,1,2,3,4,/EA)	2.0			V	5V
V_{IH2}	Input High voltage (RESET)	3.0			V	5V
I_{OL1}	Sinking Current for Output Low (P1,P2,P3,P4)	4	6		mA	5V
I_{OL2}	Sinking Current for Output Low (P0,ALE,PSEN)	8	12		mA	5V
I_{OH1}	Sourcing Current for Output High (P1,P2,P3,P4)	150	220		uA	5V
I_{OH2}	Sourcing Current for Output High (ALE,PSEN)	14	20		mA	5V
I_{IL}	Logic 0 input current (P1,2,3,4)		18	50	uA	V_{PIN}=0V
I_{TL}	Logic 1 to 0 transition current (P1,2,3,4)		270	600	uA	V_{PIN}=2V

DC Specification(3.3V MCU)

Symbol	Parameter	Specification				Test Condition
		Min.	Typ.	Max.	Unit	
V_{DD}	Operating Voltage	2.4	3.3	3.8	V	
I_{PWDN}	Power Down Current		<0.1		uA	3.3V
I_{IDLE}	Idle Current		2.0		mA	3.3V
I_{CC}	Operating Current		4 mA	15	mA	3.3V
V_{IL1}	Input low voltage (P0,1,2,3,4)			0.8	V	3.3V
V_{IL2}	Input low voltage (RESET,XTAL1)			1.5	V	3.3V
V_{IH1}	Input High voltage (P0,1,2,3,4,/EA)	2.0			V	3.3V
V_{IH2}	Input High voltage (RESET)	3.0			V	3.3V
I_{OL1}	Sinking Current for Output Low (P1,P2,P3,P4)	2.5	4		mA	3.3V
I_{OL2}	Sinking Current for Output Low (P0,ALE,PSEN)	5	8		mA	3.3V
I_{OH1}	Sourcing Current for Output High (P1,P2,P3,P4)	40	70		uA	3.3V
I_{OH2}	Sourcing Current for Output High (ALE,PSEN)	8	13		mA	3.3V
I_{IL}	Logic 0 input current (P1,2,3,4)		8	50	uA	V_{PIN}=0V
I_{TL}	Logic 1 to 0 transition current (P1,2,3,4)		110	600	uA	V_{PIN}=2V

STC89C51RC/RD+系列单片机 ISP编程 原理 注意事项



为什么有些用户下载程序不成功(在宏晶提供的下载板上)

1.可能电脑端的STC-ISP控制软件要升级，现须升级到STC-ISP-3.1
2.现在单片机端(下位机)ISP软件是3.2C,解决了少数电脑慢，通信连不上的问题.
3.运行用户程序时，可到40M/80MHz,但ISP下载程序以前的版本软件只能到33M/66MHz
4.少数客户的PLCC-44,PQFP-44转DIP-40的转换座走线过长，造成时钟振荡不稳定，下载不成功。
5.也有电脑USB电源供电不足的，可用万用表测一下，看5V部分是否在4.5V以上。
6.可能单片机内部没有ISP引导码，或PC串口波特率达不到115200，选57600试一下。
7.有些客人的笔记本电脑没有串口,用的是USB扩展串口,有些不兼容,我们可以提供经过宏晶科技认可的USB扩展串口线,人民币50元一条。

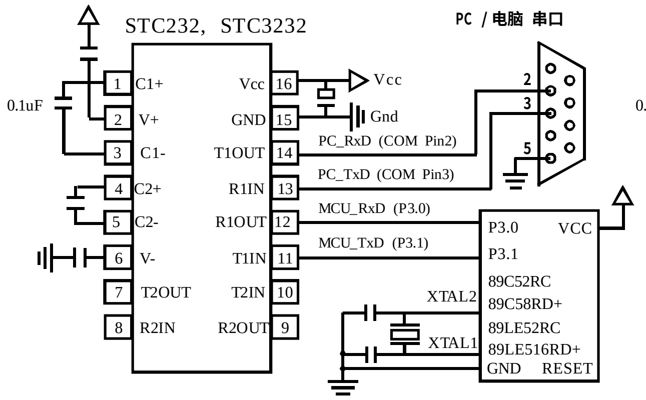
为什么有些用户下载程序不成功(在用户自己的系统上)

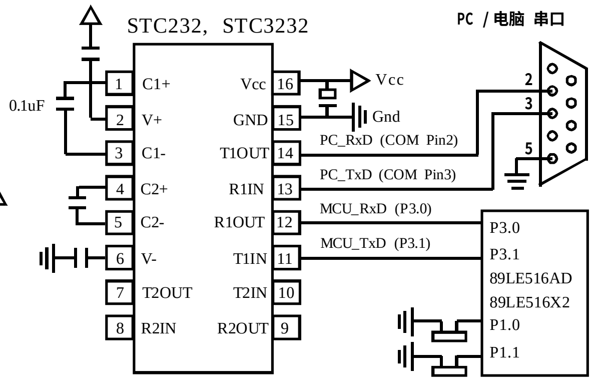
1.可能用户板上有外部看门狗，需不让其起作用，另要查时钟是否起振、复位是否正常。
2. 可能用户板上P3.0/RxD，P3.1/Txd除了接RS-232转换器外，还接了RS-485等其它电路，需要将其它电路断开,防止在ISP下载编程时受到其它电路的影响。用户系统接了RS-485/RS-232电路的，推荐在选项中选择下次冷启动时需P1.0/P1.1=0.0才判是否下载程序。

STC 89C51RC/RD+ 系列单片机在系统可编程的使用

--- 将用户代码下载进单片机内部，不用编程器

STC 单片机在线编程线路，*STC*　RS-232 转换器　　　　*STC* 单片机在线编程线路，*STC*　RS-232 转换器





上面左图适用如下型号：

STC89C51RC，　STC89C52RC，　STC89C53RC
STC89LE51RC，　STC89LE52RC，　STC89LE53RC
STC89C54RD+，　STC89C58RD+，　STC89C516RD+
STC89LE54RD+，　STC89LE58RD+，　STC89LE516RD+
STC89C516RD，　STC89C58RD　（老产品，不要选）
STC89LV516RD，　STC89LV58RD　（老产品，不要选）

上面右图适用如下型号：

STC89LE516AD,STC89LE516X2,STC89LE58AD,STC89LE54AD,STC89LE52AD,LE51AD

STC89 系列单片机大部分具有在系统可编程（ISP）特性，ISP　的好处是：省去购买通用编程器，单片机在用户系统上即可下载／烧录用户程序，而无须将单片机从已生产好的产品上拆下，再用通用编程器将程序代码烧录进单片机内部。有些程序尚未定型的产品可以一边生产，一边完善，加快了产品进入市场的速度，减小了新产品由于软件缺陷带来的风险。由于可以将程序直接下载进单片机看运行结果故也可以不用仿真器。

大部分 STC89 系列单片机在销售给用户之前已在单片机内部固化有 ISP 系统引导程序，配合 PC 端的控制程序即可将用户的程序代码下载进单片机内部，故无须编程器(速度比通用编程器快)。不要用通用编程器编程，否则有可能将单片机内部已固化的 ISP 系统引导程序擦除，造成无法使用 STC 提供的 ISP 软件下载用户的程序代码。

如何获得及使用 STC 提供的 ISP 下载工具（STC-ISP.exe 软件）：

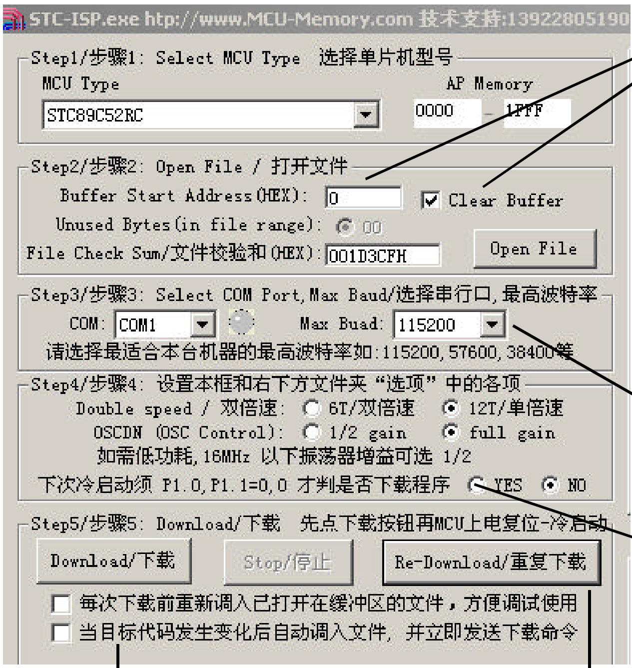
(1)．获得 STC 提供的 ISP 下载工具（软件）

　　登陆 www.MCU-Memory.com 网站，从 STC 半导体专栏下载 PC（电脑）端的 ISP 程序，然后将其自解压，再安装即可（执行 setup.exe),注意随时更新软件。

(2)．使用 STC-ISP 下载工具（软件),请随时更新，目前已到 Ver3.1 版本(2005/12/7),支持 *.Hex(Intel　16　进制格式)文件,RC/RD+ 系列单片机的底层软件版本为 Ver3.2C(旧版可更换)。
请随时注意升级 PC（电脑）端的 ISP 程序,现　Ver3.1 欢迎测试。
单片机底层软件版本为 Ver3.2C 的单片机，PC（电脑）端的 ISP 程序应用 Ver3.1 以上

(3).已经固化有 ISP 引导码,并设置为上电复位进入 ISP 的 STC89C51RC/RD+ 系列单片机出厂时就已完全加密,需要单片机内部的电放光后上电复位(冷起动)才运行系统 ISP 程序。

(4).可能用户板上 P3.0/RxD，P3.1/Txd 除了接 RS-232 转换器外，还接了 RS-485 等电路，需要将其断开。用户系统接了 RS-485 电路的，推荐在选项中选择下次冷启动时需 P1.0/P1.1=0.0 才判是否下载程序。



第一次调文件进缓冲区，要清缓冲区。
要调几个文件进缓冲区，如 EEPROM 里的数据文件需要和应用程序文件一次同时 ISP 下载编程进单片机：

除每次均要指定缓冲区起始地址外，第二次及以后不能清缓冲区
如可将要写入 EEPROM 区的数据文件调入从缓冲区 2000H/8000H 开始的地方，并不清缓冲区，然后和应用程序一起写入

用户根据实际使用效果选择限制最高通信波特率，如 57600，38400，19200

如 P3.0/P3.1 外接 RS-485/RS-232 等通信电路，建议选择如 P1.0/P1.1 不同时等于 0/0，则直接运行用户程序，跨过系统 ISP 引导程序
新的设置冷启动（彻底停电后再上电）后才生效

开发调试时，可考虑选择此项

大批量生产时使用

Step1/ 步骤 1：选择你所使用的单片机型号，如 STC89C58RD+， STC89LE516AD 等
Step2/ 步骤 2：打开文件，要烧录用户程序，必须调入用户的程序代码（*.bin， *.hex）
Step3/ 步骤 3：选择串行口，你所使用的电脑串口，如串行口 1--COM1， 串行口 2--COM2,...
有些新式笔记本电脑没有 RS-232 串行口，可买一条 USB-RS232 转接器，人民币 50 元左右。
有些 USB-RS232 转接器，不能兼容，可让宏晶帮你购买经过测试的转换器。
Step4/ 步骤 4：设置是否双倍速，双倍速选中 Double Speed 即可
STC89C51RC/RD+ 系列可以反复设置 双倍速 / 单倍速，新的设置停电后重新冷启动后才能生效
STC89LE516AD 为单倍速，STC89LE516X2 为双倍速，用户自己无法指定双倍速 / 单倍速
STC89C516RD 系列出厂时为单倍速，用户可指定设为双倍速，如想从双倍速恢复成单倍速，则需用通用编程器擦除整个晶片方可，这会将单片机内部已烧录的 ISP 引导程序擦除。一般使用缺省设置即可，无须设置。
OSCDN：单片机时钟振荡器增益降一半
选 1/2 gain 为降一半，降低 EMI；选 full gain（全增益）为正常状态。
Step5/ 步骤 5：选择 " Download/ 下载 " 按钮下载用户的程序进单片机内部，
可重复执行 Step5/ 步骤 5， 也可选择 " Re-Download/ 重复下载 " 按钮
下载时注意看提示，主要看是否要给单片机上电或复位， 下载速度比一般通用编程器快。
一般先选择 " Download/ 下载 " 按钮，然后再给单片机上电复位 (先彻底断电)，而不要先上电

关于硬件连接：

(1). MCU/ 单片机　RXD(P3.0)　---　RS-232 转换器　---　PC/ 电脑　TXD(COM Port Pin3)
(2). MCU/ 单片机　TXD(P3.1)　---　RS-232 转换器　---　PC/ 电脑　RXD(COM Port Pin2)
(3). MCU/ 单片机　GND　----------------------　PC/ 电脑　GND(COM Port Pin5)
(4). STC89LE516AD/X2 系列单片机冷启动时需要 P1.0， P1.1 = 0， 0 ；上电复位才进入 ISP 模式，下载完后释放 P1.0,P1.1， STC89LE516AD/X2 运行用户程序.
STC89C51RC/RD+ 系列单片机不需要 P1.0,P1.1 = 0， 0，但软件可选下次需要。
STC89LE516AD,STC89LE516X2,STC89LE58AD,STC89LE54AD 必需要 P1.0,P1.1 = 0， 0
(5). RS-232 转换器可选用 STC232/MAX232/SP232(4.5-5.5V),STC3232/MAX3232/SP3232(3V-5.5V).
STC232/MAX232/SP232 尽量选用 SOP 封装 (窄体,STC232ESE,STC3232ESE)。

如用户系统没有 RS-232 接口，
可使用 STC-ISP Ver 2.0B.PCB 演示板作为编程工具
可使用 STC-ISP Ver 3.0A.PCB 演示板作为编程工具

STC-ISP Ver 2.0B PCB 板可完成下载/烧录用户程序的功能。

在 STC-ISP Ver 2.0B PCB 板完成下载/烧录：

关于硬件连接：

(1.) 根据单片机的工作电压选择单片机电源电压

A. 5V 单片机,短接 JP1 的 MCU-VCC, 5V 电源管脚

B. 3V 单片机,短接 JP1 的 MCU-VCC, 3.3V 电源管脚

(2.) 根据单片机的工作电压选择复位信号(3.0A PCB 演示工具无复位信号选择)

A. 5V 单片机,短接 JP2 的 MCU-RST, 5V/MCU-RST 信号管脚

B. 3.3V 单片机,短接 JP2 的 MCU-RST, 3.3V/MCU-RST 信号管脚

(3.)连接线(宏晶提供)

A. 将一端有 9 芯连接座的插头插入 PC/电脑 RS-232 串行接口插座用于通信

B. 将同一端的 USB 插头插入 PC/电脑 USB 接口用于取电

C. 将只有一个 USB 插头的一端插入宏晶的 STC-ISP Ver 2.0B PCB 板 USB1 插座用于 RS-232 通信和供电,此时+5V Power 灯亮(D10,USB 接口有电)

(4.)其他插座不需连接

(5.)SW1 开关处于非按下状态，此时 MCU-VCC Power 灯不亮(D9)，没有给单片机通电

(6.)SW3 开关

处于非按下状态，P1.0, P1.1 = 1, 1,不短接到地。
适用于：STC89C51RC / RD+, /RD 系列单片机

处于按下状态，P1.0, P1.1 = 0, 0,短接到地。
适用于：STC89LE516AD, STC89LE58AD, STC89LE54AD, STC89LE516X2

(7.)将单片机插进 U1-Socket 锁紧座，锁紧单片机

(8.)关于软件：选择“Download/下载”(必须在给单片机上电之前让 PC 先发一串合法下载命令)

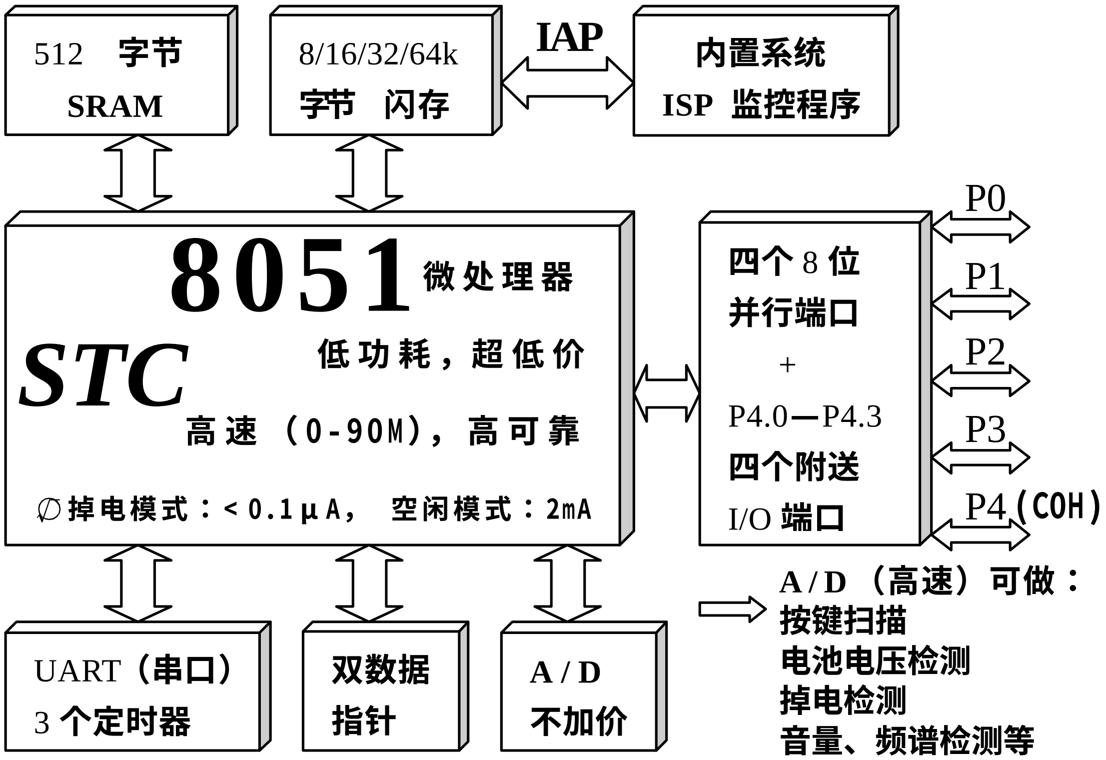
(9.)按下 SW1 开关，给单片机上电复位，此时 MCU-VCC Power 灯亮(D9)
此时 STC 单片机进入 ISP 模式(STC89C51RC/RD+系列冷启动进入 ISP)

(10.)下载成功后，再按 SW1 开关，此时 SW1 开关处于非按下状态，MCU-VCC Power 灯不亮(D9)，给单片机断电，取下单片机。

利用 STC-ISP Ver 2.0B PCB 板进行 RS-232 转换
利用 STC-ISP Ver 3.0A PCB 板进行 RS-232 转换
单片机在用户自己的板上完成下载/烧录：

1. U1-Socket 锁紧座不得插入单片机
2. 将用户系统上的电源(MCU-VCC,GND)及单片机的 P3.0/RXD,P3.1/TXD 接入转换板 CN2 插座
这样用户系统上的单片机就具备了与 PC/电脑进行通信的能力
3. 将用户系统的单片机的 P1.0,P1.1 接入转换板 CN2 插座(仅 STC89LE516AD/X2 系列需要)
4. SW3 开关处于按下状态，P1.0, P1.1 = 0, 0,短接到地。仅 STC89LE516AD/X2 系列需要
5. 关于软件：选择“Download/下载”
6. 给单片机系统上电复位(注意是从用户系统自供电，不要从电脑 USB 取电,电脑 USB 座不插)
7. 下载程序时，如用户板有外部看门狗电路，不得启动，单片机必须有正确的复位,但不能在 ISP 下载程序时被外部看门狗复位,可将外部看门狗电路 WDI 端/或 WDO 端浮空
8. 如有 RS-485 晶片连到 P3.0/Rxd,P3.1/Txd,在下载时应将其断开。

附录 A:　　STC89LE516AD 系列单片机手册



STC 具有 A/D 转换功能的单片机选型指南

型　号	最高时钟频率 Hz 1.9-3.6V	程序存储器	RAM	降低EMI	双倍速	P4口	ISP	A/D	供货
STC89LE516 AD	0-90M	64K	512	√		√	√	√	现货
STC89LE58 AD	0-90M	32K	512	√		√	√	√	现货
STC89LE54 AD	0-90M	16K	512	√		√	√	√	现货
STC89LE52 AD	0-90M	8K	512	√		√	√	√	现货
STC89LE51 AD	0-90M	4K	512	√		√	√	√	定货
STC89LE516 X2	0-90M	64K	512	√	√	√	√	√	现货

尽量选择代表宏晶最高水平的 STC12C5404AD 系列单片机，不要选择落后的 89 系列单片机

STC89LE516AD 系列单片机使用注意事项：

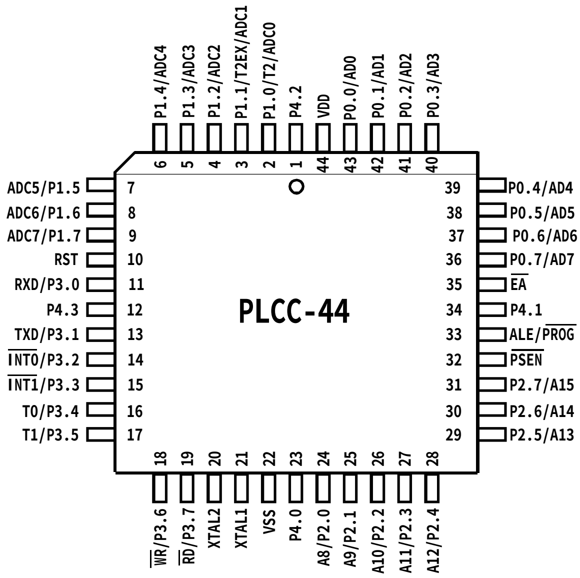
1.P1.0,P1.1 尽量留一个口不用，因为上电复位时如果 P1.0/P1.1 = 0/0，则跑 ISP 程序，而不运行用户程序
2.下载用户程序时，必须将P1.0，P1.1短接到地，发下载命令，然后再给单片机上电
3.P1口当中不做A/D转换的I/O口，尽量不用，要用，也尽量作为输入口用，不要作为输出口用
4.复位要用 MAX810R 复位电路才可靠.
5.在 XTAL2 脚串一个 200 欧姆左右的电阻.

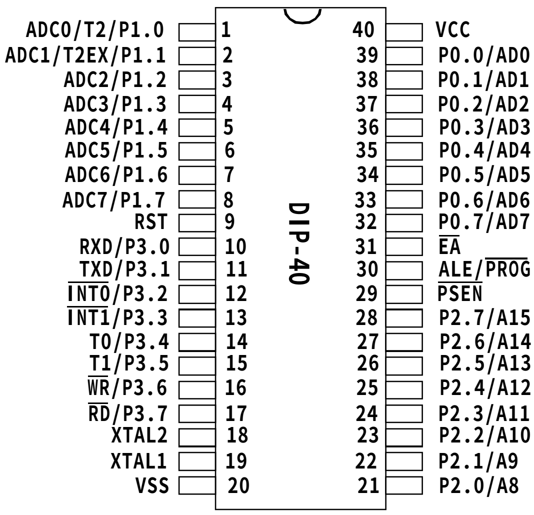
特殊功能寄存器映像 SFR Mapping

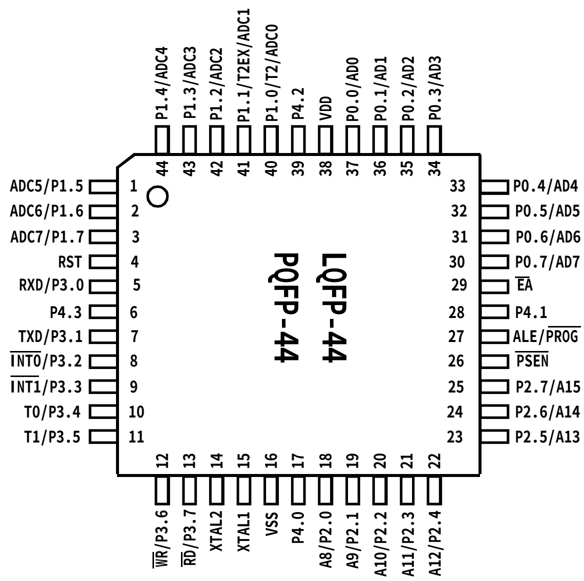
STC89LE516AD,STC89LE58AD,STC89LE54AD,STC89LE52AD,STC89LE51AD 为单倍速
因 AD 系列用户 ISP 设不了双倍速，故有一 STC89LE516AD 的双倍速版本(6T)STC89LE516X2
是同一芯片，只是出厂时就设为双倍速了。

	Bit Addressable	Non Bit Addressable							
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	
F8h									FFh
F0h	B 0000,0000								F7h
E8h									EFh
E0h	ACC 0000,0000								E7h
D8h									DFh
D0h	PSW 0000,0000								D7h
C8h	T2CON 0000,0000	T2MOD xxxx,xx00	RCAP2L 0000,0000	RCAP2H 0000,0000	TL2 0000,0000	TH2 0000,0000			CFh
C0h	P4 xxxx,1111					ADC_CONTR xxx0,0000	ADC_DATA xxxx,xxxx		C7h
B8h	IP x000,0000	SADEN 0000,0000							BFh
B0h	P3 1111,1111							IPH x000,0000	B7h
A8h	IE 0000,0000	SADDR 0000,0000							AFh
A0h	P2 1111,1111		AUXR1 xxxx,xxx0						A7h
98h	SCON 0000,0000	SBUF xxxx,xxxx							9Fh
90h	P1 1111,1111							P1_ADC_EN 0000,0000	97h
88h	TCON 0000,0000	TMOD 0000,0000	TL0 0000,0000	TL1 0000,0000	TH0 0000,0000	TH1 0000,0000	AUXR xxxx,xxx0		8Fh
80h	P0 1111,1111	SP 0000,0111	DPL 0000,0000	DPH 0000,0000				PCON 0xx1,0000	87h
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	

STC89LE516AD / X2 系列单片机 管脚图







A/D 转换典型应用线路：

按键扫描

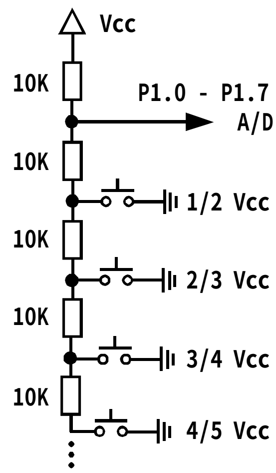
工作电压：3.6V - 2.4V， 可低至 2.4V - 1.9V

I/O 口： P0,P2,P3,P4 口可以承受 5V 输入。

P1 口不要接到 5V 系统，只能接 3.3V 系统.

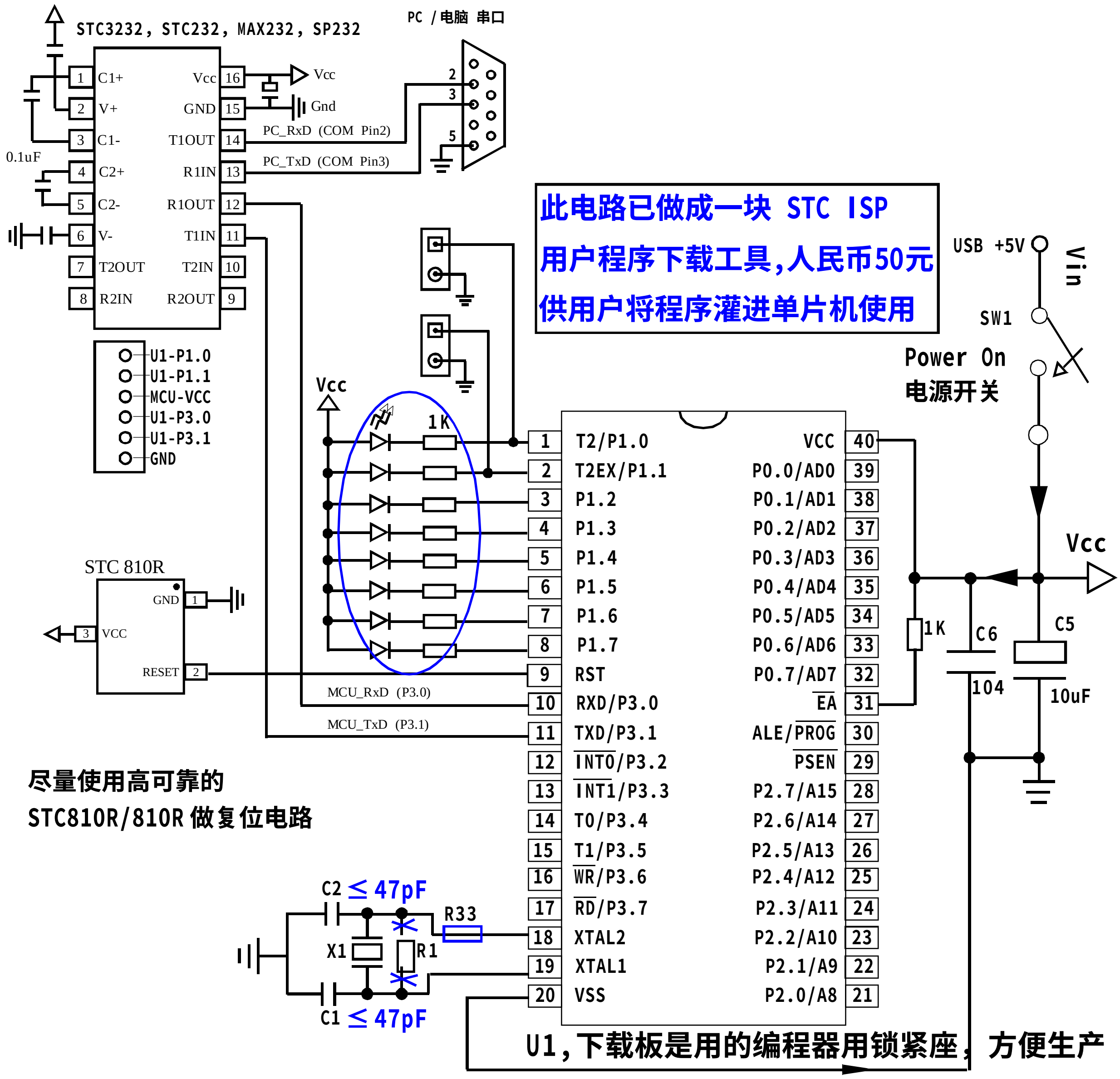
P4 口： 地址在 C0H，P4.3 - P4.0

A/D 转换在 P1 口，P1.0 - P1.7 八路。



STC89LE516AD 系列单片机 ISP 下载编程典型应用电路

STC 单片机在线编程线路， *STC* RS-232 转换器



R33 = 100 - 200 欧姆，推荐用此电阻，尽量加上								
X1	2-3MHz	4-9MHz	10-18MHz	19-26MHz	27-34MHz	35-39MHz	40-43MHz	44-48MHz
C1,C2	47pF	20pF	10pF	不用	10pF	10pF	10pF	5pF
R1	不用	不用	不用	不用	6.8K	5.1K	3.3K	3.3K

任何一种复位后，如 P1.1/P1.0 = 0,0 则进入系统 ISP 监控程序。
任何一种复位后，如 P1.1,P1.0 不同时为 0，则不进入系统 ISP 监控程序。
注意与 STC89C51RC/RD+ 系列单片机的不同,RC/RD+ 系列是冷启动进入系统 ISP 监控程序。
使用 STC89LE516AD 系列单片机时,尽量将 P1.0/P1.1 中的一个管脚空着，免得不需要进入系统 ISP 监控程序时，由于 P1.1/P1.0 = 0,0 复位后进入系统 ISP 监控程序。

1.STC89LE516AD 系列单片机扩展 AUX-RAM 的使用

STC89LE516AD/X2 系列单片机内部的 RAM 为 512 字节(256+256)，即扩展了 256 字节的 AUX-RAM，其访问方式为：

汇编语言：　(访问内部扩展的 256 字节 AUX-RAM)

```
MOVX    @Ri,  A    ;   将累加器 A 的值送至 @Ri 指向的单元, i = 0, 1
MOVX    A,    @Ri  ;   将 @Ri 指向的单元的值读到累加器 A, i = 0, 1
```

STC89LE516AD/X2 系列单片机用 " MOVX　A,　@Ri ", " MOVX　A,　@Ri " 指令访问不到外部 64K 数据空间。

汇编语言：　(访问外部 64K 数据空间)

```
MOVX    @DPTR,  A      ;   将累加器 A 的值送至 @DPTR 指向的单元
MOVX    A,      @DPTR  ;   将 @DPTR 指向的单元的值读到累加器 A
```

STC89LE516AD/X2 系列单片机用 " MOVX　A,　@DPTR ", " MOVX　A,　@DPTR " 指令访问不到内部扩展的 256 字节 AUX-RAM。

C 语言：

用 pdata　声明的变量访问单片机内部扩展的 256 字节 AUX-RAM

用 xdata　声明的变量访问单片机外部 64K 数据空间

2.双数据指针　及　AUXR1 寄存器

AUXR1	A2h	Auxiliary Register 1	-	-	-	-	-	-	-	DPS	xxxx,xxx0

DPS = 0 时选择 DPTR0,　DPS = 1 时选择 DPTR1

可以用 " INC AUXR1 " 快速切换 DPTR0 / DPTR1

3.禁止 ALE 输出　及　AUXR1 寄存器

AUXR	8Eh	Auxiliary Register 0	-	-	-	-	-	-	-	ALEOFF	xxxx,xxx0

ALEOFF = 0, ALE 信号正常输出。

ALEOFF = 1, 禁止 ALE 信号输出。但在访问外部数据空间及外部程序空间时有信号输出。

4.中断　及　中断优先级控制寄存器 IP / IPH

中断与普通 8052 完全兼容，优先级可设为 4 级，通过增加的 IPH 寄存器

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
IE	A8h	Interrupt Enable	EA	-	ET2	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	-	-	PT2	PS	PT1	PX1	PT0	PX0	xx00,0000
IPH	B7h	Interrupt Priority High	-	-	PT2H	PSH	PT1H	PX1H	PT0H	PX0H	xx00,0000

Interrupt Source 中断源	Vector Address 中断向量地址	Polling Sequence 中断查询次序	中断优先级设置	优先级0 最低	优先级1	优先级2	优先级3 最高	Interrupt Request 中断请求
/INT0	0003H	1	PX0H,PX0	0,0	0,1	1,0	1,1	IE0
Timer 0	000BH	2	PT0H,PT0	0,0	0,1	1,0	1,1	TF0
/INT1	0013H	3	PX1H,PX1	0,0	0,1	1,0	1,1	IE1
Timer 1	001BH	4	PT1H,PT1	0,0	0,1	1,0	1,1	TF1
UART	0023H	5	PSH, PS	0,0	0,1	1,0	1,1	RI + TI
Timer 2	002BH	6	PT2H,PT2	0,0	0,1	1,0	1,1	TF2 + EXF2

4.A/D 及 A/D 转换寄存器 ADC_CONTR/ADC_DATA/P1_ADC_EN

STC89LE516AD/X2 在 P1 口，有 8 位精度的高速 A/D 转换器，P1.7 - P1.0 共 8 路电压输入型，可做按键扫描，电池电压检测，频谱检测等。17 个机器周期可完成一次转换，时钟在 40MHz 以下时。

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P1_ADC_EN	97h	允许P1.x成为A/D口	ADC_P17	ADC_P16	ADC_P15	ADC_P14	ADC_P13	ADC_P12	ADC_P11	ADC_P10	0000,0000
ADC_CONTR	C5h	A/D 转换控制寄存器	-	-	-	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	xxx0,0000
ADC_DATA	C6h	A/D 转换结果寄存器	-	-	-	-	-	-	-	-	0000,0000

P1_ADC_EN 特殊功能寄存器： P1.x 作为 A/D 转换输入通道来用允许特殊功能寄存器

允许P1.x成为A/D口	ADC_P17	ADC_P16	ADC_P15	ADC_P14	ADC_P13	ADC_P12	ADC_P11	ADC_P10	0000,0000

相应位为 “ 1 ” 时，对应的 P1.x 口作为 A/D 转换使用,内部上拉电阻自动断开

ADC_CONTR 特殊功能寄存器： A/D 转换控制特殊功能寄存器

A/D 转换控制寄存器	-	-	-	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	xxx0,0000

CHS2 / CHS1 / CHS0：模拟输入通道选择，CHS2 / CHS1 / CHS0

CHS2	CHS1	CHS0	Analog Input Channel Select 模拟输入通道选择
0	0	0	选择 P1.0 作为 A/D 输入来用
0	0	1	选择 P1.1 作为 A/D 输入来用
0	1	0	选择 P1.2 作为 A/D 输入来用
0	1	1	选择 P1.3 作为 A/D 输入来用
1	0	0	选择 P1.4 作为 A/D 输入来用
1	0	1	选择 P1.5 作为 A/D 输入来用
1	1	0	选择 P1.6 作为 A/D 输入来用
1	1	1	选择 P1.7 作为 A/D 输入来用

ADC_START: 模拟/数字转换(ADC)启动控制位，设置为 “ 1 ” 时，开始转换

ADC_FLAG: 模拟/数字转换结束标志位,当 A/D 转换完成后，ADC_FLAG = 1。

ADC_DATA 特殊功能寄存器： A/D 转换结果特殊功能寄存器

A/D 转换结果寄存器	-	-	-	-	-	-	-	-	0000,0000

模拟/数字转换结果计算公式如下： 结果 = 256 x Vin / Vcc

Vin 为模拟输入通道输入电压，Vcc 为单片机实际工作电压，用单片机工作电压作为模拟参考电压。

一个完整的 A/D 转换测试程序

```
;/* ---  STC International Limited ---------------------------------------- */
;/* ---  宏晶科技 姚永平  设计 2006/1/6  V1.0 ---------------------------- */
;/* ---  演示 STC89LE516AD/X2 系列 MCU A/D 转换功能 ---------------------- */
;/* ---  Mobile: 13922805190 --------------------------------------------- */
;/* ---  Tel: 0755-82948409    Fax: 0755-82944243------------------------ */
;/* ---  Web: www.mcu-memory.com ----------------------------------------- */
;/* ---  本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过 ---------- */
;/* ---  如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 --- */
;/* ---  如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序 --- */

//   ADC DEMO 程序演示 STC89LE516AD/X2 系列 MCU 的 A/D 转换功能。 时钟   11.0592MHz
//  转换结果以 16 进制形式输出到串行口，可以用串行口调试程序观察输出结果。
#include <reg52.H>
#include <intrins.H>
// 定义与 ADC 有关的特殊功能寄存器
sfr  P1_ADC_EN     =    0x97;              // A/D 转换功能允许寄存器
sfr  ADC_CONTR     =    0xC5;             // A/D 转换控制寄存器
sfr  ADC_DATA      =    0xC6;             // A/D 转换结果寄存器
typedef    unsigned char    INT8U;
typedef    unsigned int    INT16U;
void delay(INT8U delay_time)            // 延时函数
{
    INT8U      n;
    INT16U     m;
    for (n=0; n<delay_time; n++)
    {
        for (m=0; m<10000; m++);
    }
}
void initiate_RS232 (void)          // 串口初始化
{
    ES = 0;                         // 禁止串口中断
    SCON = 0x50;                    // 0101,0000  8 位数据位， 无奇偶校验
    T2CON = 0x34;                     // 0011，0100， 由 T2 作为波特率发生器
    RCAP2H = 0xFF;                   // 时钟 11.0592MHz， 9600 波特率
    RCAP2L = 0xDB;

    ES = 1;                         // 允许串口中断
}
void Send_Byte(INT8U one_byte)      // 发送一个字节
{
    TI = 0;                         // 清零串口发送中断标志
    SBUF = one_byte;
    while (TI == 0);
    TI = 0;                          // 清零串口发送中断标志
}
```

```
INT8U get_AD_result(INT8U channel)
{
    INT8U AD_finished     =    0;            // 存储 A/D 转换标志
    ADC_DATA    =      0;
    ADC_CONTR  = channel;                   //选择 A/D 当前通道
    delay(1);                               //使输入电压达到稳定
    ADC_CONTR |= 0x08;                      //0000,1000 令 ADC_START = 1, 启动A/D转换
    AD_finished = 0;
    while ( AD_finished == 0 )              //等待A/D转换结束
    {
        AD_finished = (ADC_CONTR & 0x10);  //0001,0000, ADC_FLAG ==1 测试A/D转换结束否
    }
    ADC_CONTR &= 0xF7;                     //1111,0111 令 ADC_START = 0, 关闭A/D转换,
    return (ADC_DATA);                    // 返回 A/D 转换结果
}
void main()
{
    initiate_RS232();
    P1     =     P1    |     0x63;    // 0110,0011,要设置为 A/D 转换的P1.x口,先设为高
    P1_ADC_EN = 0x63;       //0110,0011, P1 的P1.0,P1.1,P1.5,P1.6设置为 A/D 转换输入脚
                            //断开P1.0,P1.1,P1.5,P1.6内部上拉电阻
    while(1)
    {
        Send_Byte(get_AD_result(0));  //P1.0 为 A/D 当前通道, 测量并发送结果
        delay(0x200);

        Send_Byte(get_AD_result(1));  //P1.1 为 A/D 当前通道, 测量并发送结果
        delay(0x200);

        Send_Byte(get_AD_result(5));  //P1.5 为 A/D 当前通道, 测量并发送结果
        delay(0x200);

        Send_Byte(get_AD_result(6));  //P1.6 为 A/D 当前通道, 测量并发送结果
        delay(0x200);

        Send_Byte(0);               //连续发送 4 个 00H, 便于观察输出显示
        Send_Byte(0);
        Send_Byte(0);
        Send_Byte(0);

        delay(0x200);               //延时
        delay(0x200);
        delay(0x200);
        delay(0x200);
        delay(0x200);
        delay(0x200);
    }
}
```

附录B:为什么少数用户的普通8051程序烧录后，不能运行

1. 增强型的STC89C58RD+，STC89C52RC系列单片机内部扩展了AUX-RAM.

型号	内部扩展RAM	占外部64K数据空间	MOVX @DPTR / xdata	MOVX @Ri / pdata
STC89C51RC	256 字节	0000H - 00FFH	地址>=100H,才访问外部	只访问内部扩展RAM
STC89LE51RC	256 字节	0000H - 00FFH	地址>=100H,才访问外部	只访问内部扩展RAM
STC89C52RC	256 字节	0000H - 00FFH	地址>=100H,才访问外部	只访问内部扩展RAM
STC89LE52RC	256 字节	0000H - 00FFH	地址>=100H,才访问外部	只访问内部扩展RAM
STC89C53RC	256 字节	0000H - 00FFH	地址>=100H,才访问外部	只访问内部扩展RAM
STC89LE53RC	256 字节	0000H - 00FFH	地址>=100H,才访问外部	只访问内部扩展RAM
STC89C54RD+	1024 字节	0000H - 03FFH	地址>=400H,才访问外部	只访问内部扩展RAM
STC89LE54RD+	1024 字节	0000H - 03FFH	地址>=400H,才访问外部	只访问内部扩展RAM
STC89C58RD+	1024 字节	0000H - 03FFH	地址>=400H,才访问外部	只访问内部扩展RAM
STC89LE58RD+	1024 字节	0000H - 03FFH	地址>=400H,才访问外部	只访问内部扩展RAM
STC89C516RD+	1024 字节	0000H - 03FFH	地址>=400H,才访问外部	只访问内部扩展RAM
STC89LE516RD+	1024 字节	0000H - 03FFH	地址>=400H,才访问外部	只访问内部扩展RAM
STC89LE516AD	256 字节		0000-FFFFH,均访问外部	只访问内部扩展RAM
STC89LE516X2	256 字节		0000-FFFFH,均访问外部	只访问内部扩展RAM

STC89C52RC / STC89C58RD+系列

如果用户的单片机 P0 / P2 口是作为普通I/O口(输入/输出口)用的，无冲突

如果用户的单片机 P0 / P2 口是作为总线扩展用，外部扩展器件的地址在内部扩展RAM之上，无冲突

如果用户的单片机 P0 / P2 口是作为总线扩展用的，访问外部扩展器件的地址在内部扩展RAM的范围内，则访问的是内部扩展的AUX-RAM,所以有些系统用户要禁止内部扩展RAM

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
AUXR	8Eh	Auxiliary Register 0	-	-	-	-	-	-	EXTRAM	ALEOFF	xxxx,xx00

将AUXR寄存器的EXTRAM设为1，即可禁止内部扩展的AUX-RAM，使之与标准8052一样。

STC89LE516AD / STC89LE516X2系列AUXR寄存器无EXTRAM控制位，通过寻址方式区分，

“MOVX @Ri ”固定访问内部扩展RAM，

“MOVX， @DPTR ”固定访问外部扩展64K数据空间，详见相应部分说明。

2. 复位电路

电阻/电容复位的值为 电阻 = 10k， 电容 = 10uF。如为用外部专用芯片复位，RESET管脚(复位脚)不要加任何上拉/下拉电阻。详见相应部分说明

附录 C: STC89C51 RC / RD+ 系列 单片机 ISP (DIY)

自己动手写自己的 ISP,写自己的远程升级程序还需了解的信息,只提供给策略性伙伴

用户程序区空间和 ISP 程序区空间:

型号	用户应用程序区	ISP 引导区空间(ISP Space)			
	AP Space 地址	0 K	1K(ISP/IAP)	2 K(ISP/IAP)	4K(ISP/IAP)
STC89C51RC	0000H - 0FFFH	目前版本禁止ISP/IAP操作	3800H - 3BFFH	3400H - 3BFFH	2C00H - 3BFFH
STC89LE51RC	0000H - 0FFFH		3800H - 3BFFH	3400H - 3BFFH	2C00H - 3BFFH
STC89C52RC	0000H - 1FFFH		3800H - 3BFFH	3400H - 3BFFH	2C00H - 3BFFH
STC89LE52RC	0000H - 1FFFH		3800H - 3BFFH	3400H - 3BFFH	2C00H - 3BFFH

型号	用户应用程序区	ISP 引导区空间(ISP Space)			
	AP Space 地址	0 K	1K(ISP/IAP)	2 K(ISP/IAP)	4K(ISP/IAP)
STC89C53RC	0000H - 3BFFH	目前版本禁止ISP/IAP操作			
STC89LE53RC	0000H - 3BFFH				
STC89C53RC	0000H - 37FFH		3800H - 3BFFH		
STC89LE53RC	0000H - 37FFH		3800H - 3BFFH		
STC89C53RC	0000H - 33FFH			3400H - 3BFFH	
STC89LE53RC	0000H - 33FFH			3400H - 3BFFH	
STC89C53RC	0000H - 2BFFH				2C00H - 3BFFH
STC89LE53RC	0000H - 2BFFH				2C00H - 3BFFH

型号	用户应用程序区	ISP 引导区空间(ISP Space),从ISP区启动MCU才有IAP功能			
	AP Space 地址	0 K	1K(ISP/IAP)	2 K(ISP/IAP)	4K(ISP/IAP)
STC89C54RD+	0000H - 3FFFH	目前版本禁止ISP/IAP操作	F800H - FBFFH	F400H - FBFFH	EC00H - FBFFH
STC89LE54RD+	0000H - 3FFFH		F800H - FBFFH	F400H - FBFFH	EC00H - FBFFH
STC89C58RD+	0000H - 7FFFH		F800H - FBFFH	F400H - FBFFH	EC00H - FBFFH
STC89LE58RD+	0000H - 7FFFH		F800H - FBFFH	F400H - FBFFH	EC00H - FBFFH

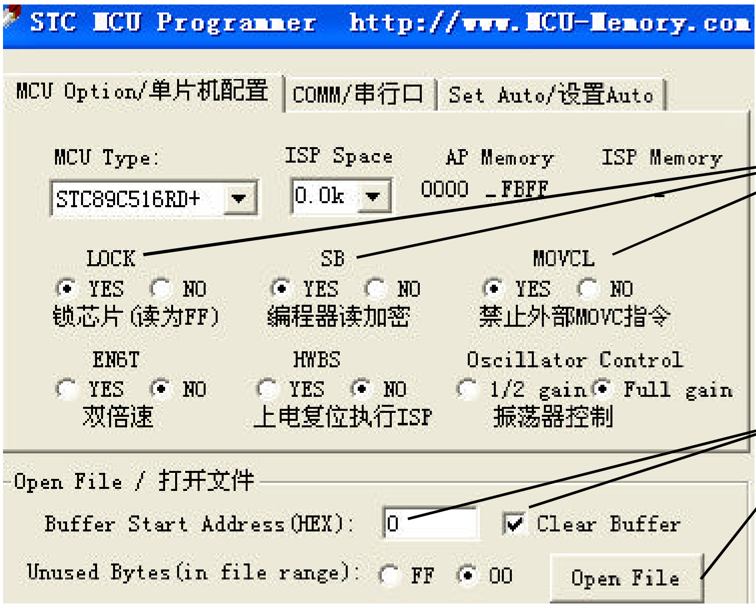
型号	用户应用程序区	ISP 引导区空间(ISP Space),从ISP区启动MCU才有IAP功能			
	AP Space 地址	0 K	1K(ISP/IAP)	2 K(ISP/IAP)	4K(ISP/IAP)
STC89C516RD+	0000H - FBFFH	目前版本禁止ISP/IAP操作			
STC89LE516RD+	0000H - FBFFH				
STC89C516RD+	0000H - F7FFH		F800H - FBFFH		
STC89LE516RD+	0000H - F7FFH		F800H - FBFFH		
STC89C516RD+	0000H - F3FFH			F400H - FBFFH	
STC89LE516RD+	0000H - F3FFH			F400H - FBFFH	
STC89C516RD+	0000H - EBFFH				EC00H - FBFFH
STC89LE516RD+	0000H - EBFFH				EC00H - FBFFH

动手写自己的 ISP 还必需要的工具,只提供给策略性伙伴

-----STC89C51RC / RD+ 系列单片机专用烧录器（200 元）

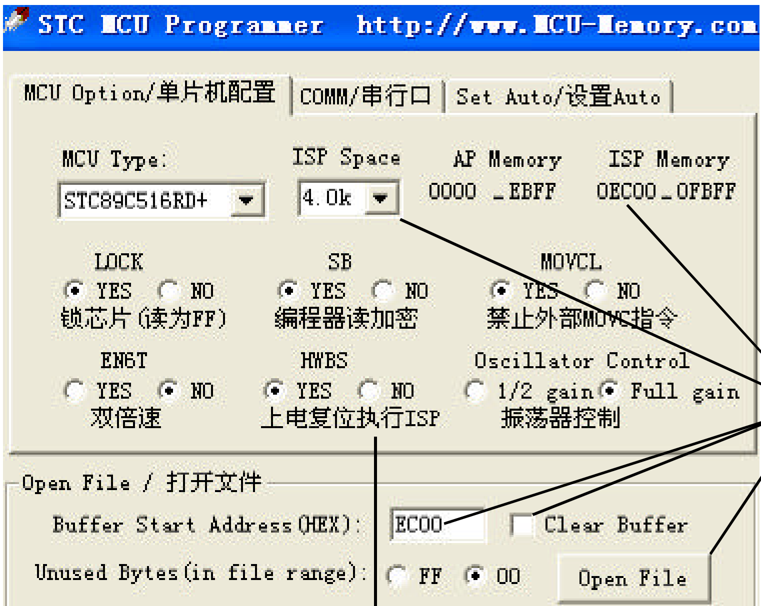
STC89C51RC / RD+ 系列单片机出厂时一般都固化有 ISP 引导码程序，如只烧录普通的用户程序，则只需将 P3.0/P3.1 经过 RS-232 转换器连到电脑的 RS-232 串口，通过 STC-ISP 下载软件打开用户程序下载就可以了。如果不用 STC 的 ISP 程序而要编写自己的 ISP 程序，
则需要 STC89C51RC / RD+ 系列单片机专用烧录器，才可将用户的 ISP 程序烧录进单片机内部,软件使用 STC MCU Programmer 编程工具。

方式一：用户主应用程序和 ISP 程序一起烧录，上电复位后程序从 ISP 区开始运行



一. 三个加密项全部加密

二. 缓冲区从 0 开始，清缓冲区调入用户主应用程序



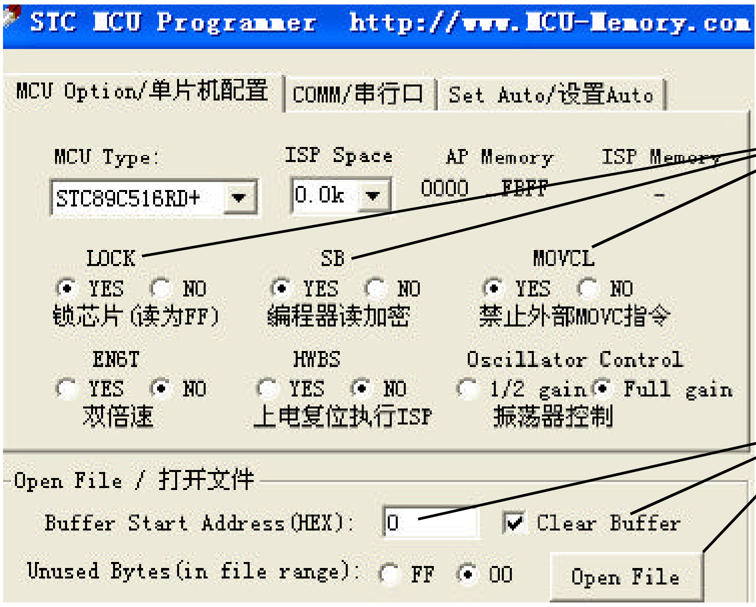
三. 选定 ISP 空间从那里开始，不清缓冲区调入 ISP 程序,保留已调入的程序，用户 ISP 空间可选 0/1K/2K/4K

四. 选择从 ISP 区启动

五. 烧录程序

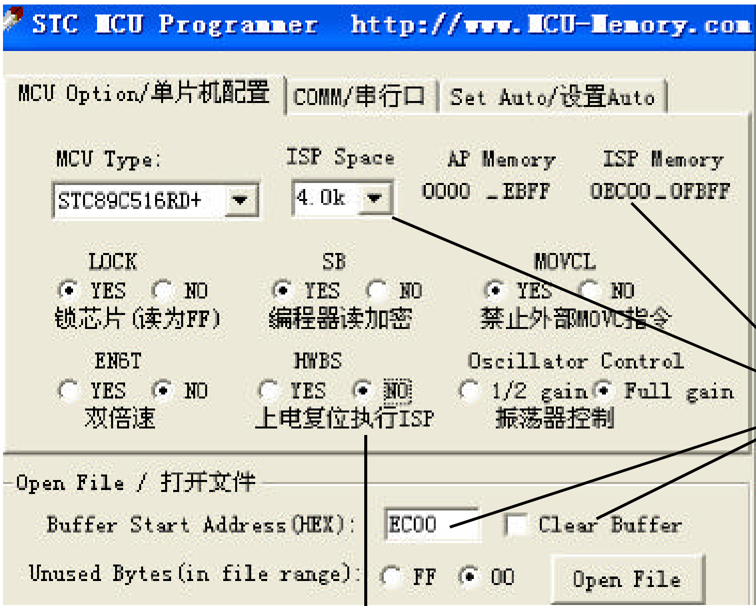
也可直接调入 ISP 程序至 ISP 区，选择从 ISP 程序区启动，再烧录。用户主应用程序由用户自己编的 ISP 程序下载。还可以把 ISP 区当成用户程序区，把余下的空间当成 Data Flash 数据空间。如 STC89C516RD+,4K ISP 区当用户程序使用，余下 59K 当 EEPROM 用。

方式二：用户主应用程序和 ISP 程序一起烧录，上电复位后程序从用户主应用程序区开始运行(但单片机 ISP 空间大小必须要选择 1K/2K/4K，否则只能当成无 ISP/IAP 的单片机使用)



一．三个加密项全部加密

二．缓冲区从 0 开始，清缓冲区调入用户主应用程序



三．选定 ISP 空间从那里开始，不清缓冲区调入用户自己的 ISP 程序，保留已调入的程序

四．选择从用户主应用程序区启动

五．烧录程序

已加过密的单片机，或已设置过 ISP 空间为 1k/2k/4k，上电复位从 ISP 空间启动的单片机，重新烧录程序时，要先将整个芯片擦除，再停电(单片机电放光后)，再上电，再重烧程序，才会写对。擦除整个芯片后，重新允许读出芯片内容的状态，是停电后再上电才能生效。

用户写自己的 ISP 程序的格式：

```
ISP_DATA     EQU   0E2H ;申明 ISP_DATA 寄存器地址，或 ISP_DATA   DATA  0E2H
ISP_ADDRH    EQU   0E3H ;申明 ISP_ADDRH 寄存器地址，或用  DATA   申明
ISP_ADDRL    EQU   0E4H ;申明 ISP_ADDRL 寄存器地址，或用  DATA   申明
ISP_CMD      EQU   0E5H ;申明 ISP_CMD 寄存器地址，   或用  DATA   申明
ISP_TRIG     EQU   0E6H ;申明 ISP_TRIG 寄存器地址，   或用  DATA   申明
ISP_CONTR    EQU   0E7H ;申明 ISP_CONTR 寄存器地址，  或用  DATA   申明
ORG          0000H;     ISP 程序的入口地址，逻辑上也是从“0”开始
LJMP         INIT_MCU
ORG          0003H;   ISP 程序的外部中断 0 入口地址，逻辑上是从“0003H”开始
LJMP         INT_0
ORG          000BH;   ISP 程序定时器 0 中断入口地址，逻辑上是从“000BH”开始
LJMP         TIMER_0
ORG          0023H;   ISP 程序的串口中断入口地址，逻辑上也是从“0023H”开始
LJMP         Serial_Port_INT
ORG         0050H;      初始化单片机
INIT_MCU:
      MOV        SP,   #0E0H;
      ......
      ......
;收到外部命令要更新‘用户主应用程序区”(AP区)
      ......
UPDATE_USER_AP_CODE:
      ......  ; 更新用户程序区(AP),用 ISP/IAP 对用户 AP 区操作，物理地址从 0 开始
From_ISP_to_AP_Soft_Reset:
      MOV        ISP_CONTR，#00100000B;更新完用户程序区后,从 ISP 区软复位到 AP 区
      END   ASM
;将以上 ISP程序烧录到单片机的 ISP区就可已了。
---------------------------------------------------
也可以在用户程序区收到命令后，从AP区软复位到 ISP区。
From_AP_to_ISP_Soft_Reset:
      MOV        ISP_CONTR，#01100000B;在 AP 区收到命令后，从 AP 区软复位到 ISP 区
```

附录 D： 如何实现运行中自定义下载,无仿真器时方便调试

自定义下载原理：STC-ISP.exe 软件

1.STC-ISP.exe 软件,可由用户设置，按 UART/RS-232 的格式向用户程序发送命令

波特率： 38400bps， 19200bps， 9600bps， 4800bps， 2400bps， 1200bps 等

奇偶校验位： 无， 偶校验， 奇校验

数据位几位： 8 位，7 位，6 位，5 位， 等

停止位几位： 1 位， 1.5 位，2 位，等



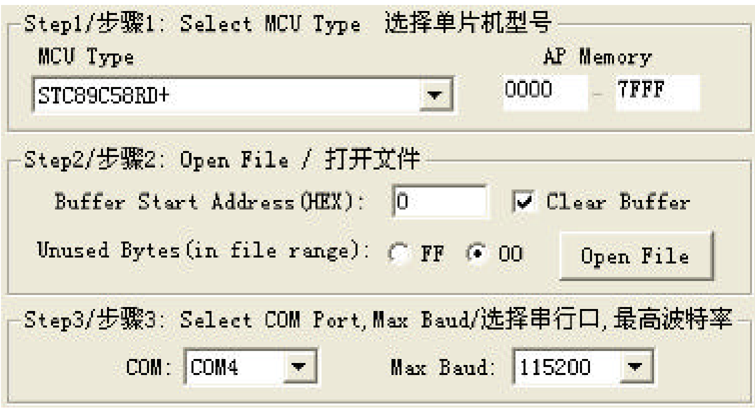
2.需向用户程序发送的命令用户可在上图自定义下载命令输入窗口中输入(HEX)
命令之间建议加一个空格，也可不加，STC-ISP.exe 会处理，上例为发送自定义命令
12H,34H,56H,78H,0ABH,0CDH,0EFH， 12H,34H,56H,78H,0ABH,0CDH,0EFH

3.用户程序中应加入收到自定义下载命令，延时一秒，软复位到系统 ISP 程序区的程序

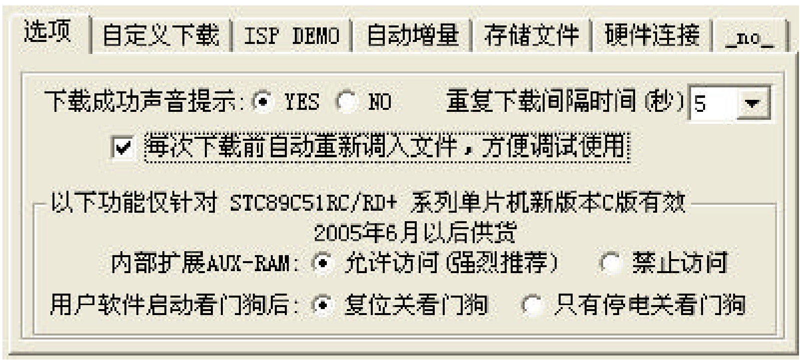
4.将以上含有接收自定义下载命令的用户程序先用老方法下载进 STC 单片机内部：
STC89C51RC,STC89C52RC,STC89C53RC,STC89C54RD+,STC89C58RD+,STC89C516RD+
STC89LE52RC,STC89LE53RC,STC89LE54RD+,STC89LE58RD+,STC89LE516RD+

这样以上型号的 STC 单片机就具有了“不停电 / 运行中 / 自定义下载功能”

6.以下选好型号，打开文件...,在自定义下载中设置相关选项，选择“发送”即可，



STC-ISP.exe 在“发送”完用户自定义下载命令后，就会转去调用老的那一套下载命令，而不管用户单片机程序收到命令没有。不过这个转换有些电脑有时需要将近 1S 的时间，所以用户应用程序要延时 1S，否则系统 ISP 程序收不到下载命令，又会回到用户应用程序。

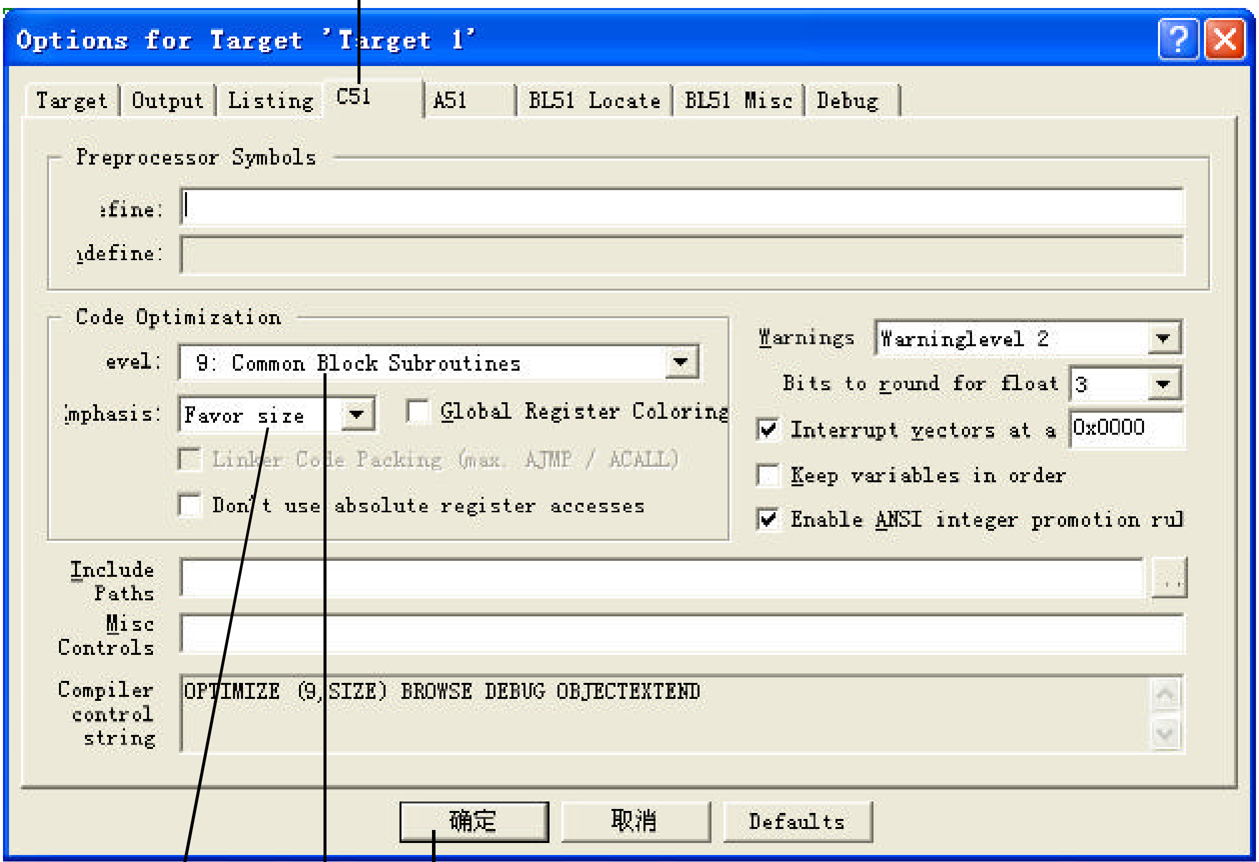


7.调试程序时，还可以在选项中选择“每次下载前自动重新调入文件”，这样你每次修改原文件并从新编译 / 汇编后生成的*.hex,*.bin 文件就不要再次手工调入了，生产时不要用。

附录 E: Keil C51 高级语言编程的软件如何减少代码长度

1.在“Project”菜单中选择“Options for Target”

2.在“Options for Target”中选择“C51”



3.选择按空间大小， 9级优化程序

4.重新编译程序即可。

附录F:(写给用STC89C51RC/RD+系列单片机做仿真器的朋友)
STC89C51 RC / RD+ 系列 单片机 做仿真器须知

1.对Flash的IAP字节读/字节编程/扇区擦除等待机器周期

设置等待时间			CPU 等待时间(机器周期)			
WT2	WT1	WT0	Read	Program	Sector Erase	Recommended System Clock
0	1	1	6	30	5471	5MHz
0	1	0	11	60	10942	10MHz
0	0	1	22	120	21885	20MHz
0	0	0	43	240	43769	40MHz

512Byte/Sector

2.程序在ISP区可对用户AP区进行IAP读/编程/扇区擦除

3.程序在用户AP区不可对ISP区进行IAP操作。

4.程序在ISP/AP区可对Data Flash区进行IAP操作。

5.可从ISP区软复位到AP区，也可从AP区软复位到ISP区。

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
ISP_DATA	E2h	ISP/IAP Flash Data Register									1111,1111
ISP_ADDRH	E3h	ISP/IAP Flash Address High									0000,0000
ISP_ADDRL	E4h	ISP/IAP Flash Address Low									0000,0000
ISP_CMD	E5h	ISP/IAP Flash Command Register	-	-	-	-	-	MS2	MS1	MS0	xxxx,x000
ISP_TRIG	E6h	ISP/IAP Flash Command Trigger									xxxx,xxxx
ISP_CONTR	E7h	ISP/IAP Control Register	ISPEN	SWBS	SWRST	-	-	WT2	WT1	WT0	000x,x000

```
;从用户应用程序区(AP区)软件复位并切换到ISP程序区开始执行程序
    MOV     ISP_CONTR,#01100000B ;SWBS = 1(选择ISP区), SWRST = 1(软复位)
;从ISP程序区软件复位并切换到用户应用程序区(AP区)开始执行程序
    MOV     ISP_CONTR,#00100000B ;SWBS = 0(选择AP区), SWRST = 1(软复位)
```

软复位后所有的寄存器复位，可在软复位前将需保存的SFR的值，保存在RAM或DATA Flash中，建议采用STC89C58RD+设计。

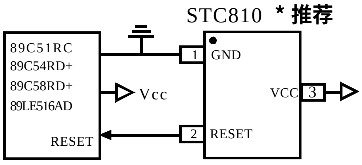
STC89C58RD+50-C-PI:　独立32k字节用户程序区，独立可选0/1/2/4k字节系统ISP区。
　　独立27k字节数据Flash区

只有选择1/2/4k字节ISP区,并选择上电复位从ISP区启动，晶片IAP功能才起作用

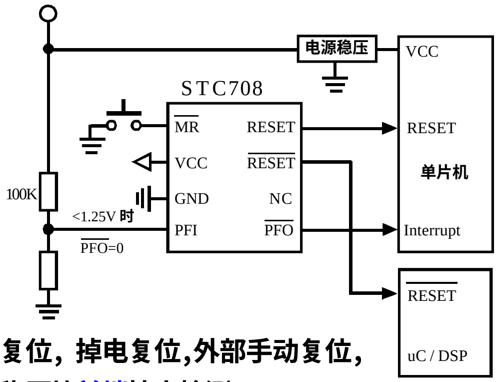
工作电压：5.5v - 3.4v， 不要过低，否则工作不了。5V串2个1N4001二极管是3.6V

附录F： 典型MCU/DSP/uC 复位、电源监控、外部看门狗专用电路

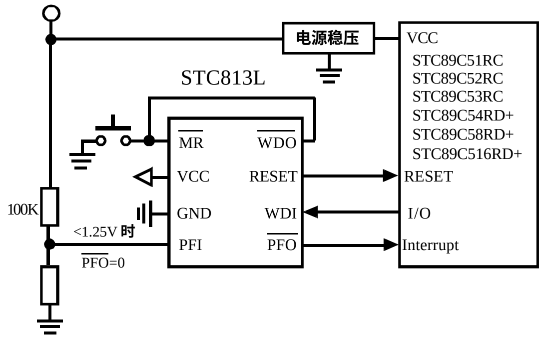
1.高电平复位信号输出



上电复位，掉电复位,人民币1元

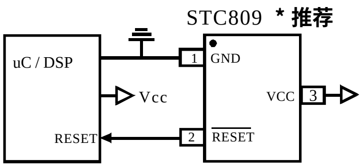


上电复位，掉电复位,外部手动复位,
电源稳压块前端掉电检测 ，
高 / 低电平 2 路复位信号输出

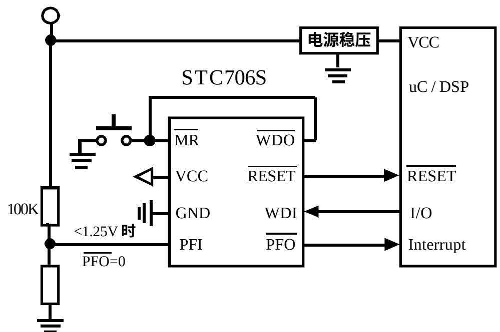


上电复位，掉电复位,外部手动复位,
电源稳压块前端掉电检测,
外部硬件看门狗

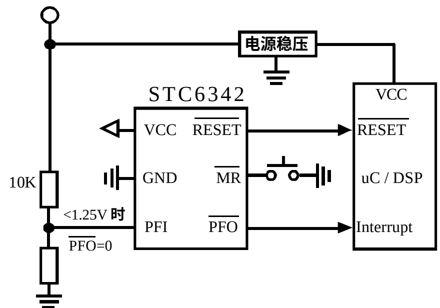
2.低电平复位信号输出



上电复位，掉电复位,人民币1元



上电复位，掉电复位,外部手动复位,
电源稳压块前端掉电检测,
外部硬件看门狗



上电复位，掉电复位,外部手动复位,稳压块前端掉电检测

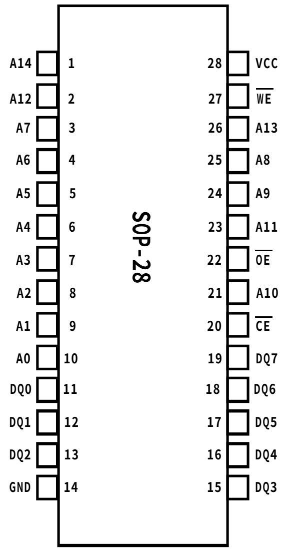
使用外部专用复位电路的好处：

1.确保上电时，在用户设定的电源电压之上，时钟振荡稳定后，单片机才开始工作
2.确保掉电时，在用户设定的电源电压之下，立即让单片机复位，以免单片机误动作
3.具有电源稳压块前端掉电检测的专用复位电路,确保掉电前有充分的时间保存数据
4.复位门槛电压可选： L:4.63V; M:4.38V; J: 4.00V; T:3.08V; S:2.93V; R:2.63V

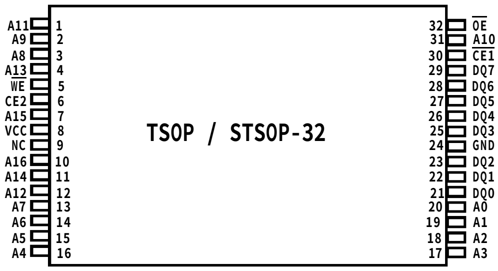
附录 H:　　STC 高性能 SRAM 选型一览表

型号	容量	工作电压	温度	速度	推荐封状	供货
STC62WV256	32K x 8	2.4 - 5.5V	-40°C~85°C	70nS	SOP/TSOP/DIP	订货
STC62WV1024	128K x 8	2.4 - 5.5V	-40°C~85°C	70nS	SOP/STSOP/TSOP	订货
STC62WV2568	256K x 8	2.4 - 5.5V	-40°C~85°C	70nS	STSOP-32	订货
STC62WV5128	512K x 8	2.4 - 5.5V	-40°C~85°C	70nS	STSOP/SOP-32	订货
STC62WV1M8	1M x 8	2.4 - 5.5V	-40°C~85°C	70nS	TSOP2-44	订货
STC62WV6416	64K x 8	2.4 - 5.5V	-40°C~85°C	70nS	TSOP2-44	订货
STC62WV12816	128K x 16	2.4 - 5.5V	-40°C~85°C	70nS	TSOP2-44	订货
STC62LV12816	128K x 16	2.4 - 3.6V	-40°C~85°C	70nS	TSOP2-44	订货
STC62WV25616	256K x 16	2.4 - 5.5V	-40°C~85°C	70nS	TSOP2-44	订货
STC62WV51216	512K x 16	2.4 - 5.5V	-40°C~85°C	70nS	TSOP2-44	订货

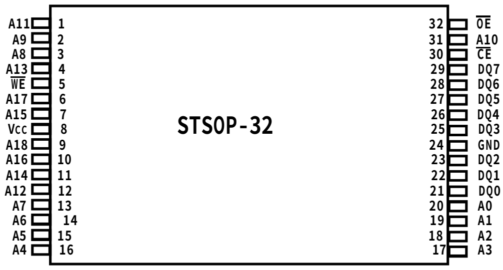
STC62WV256



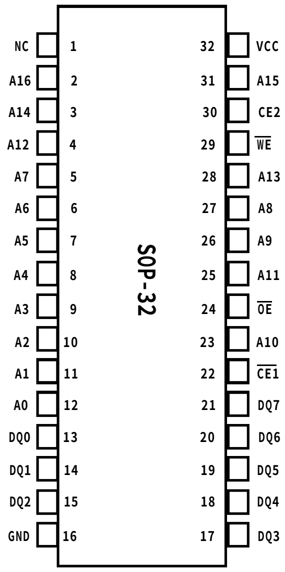
STC62WV1024



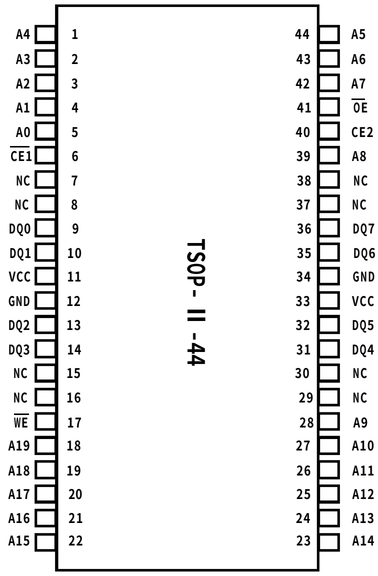
STC62WV5128



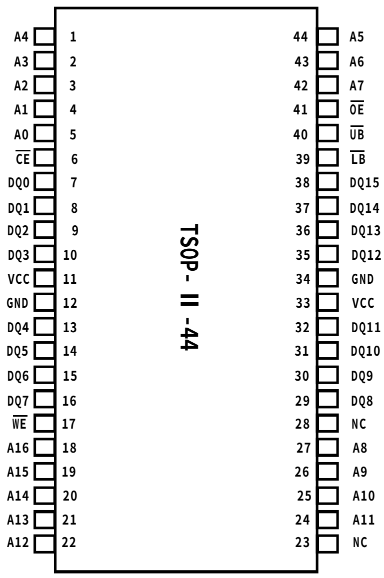
STC62WV1024



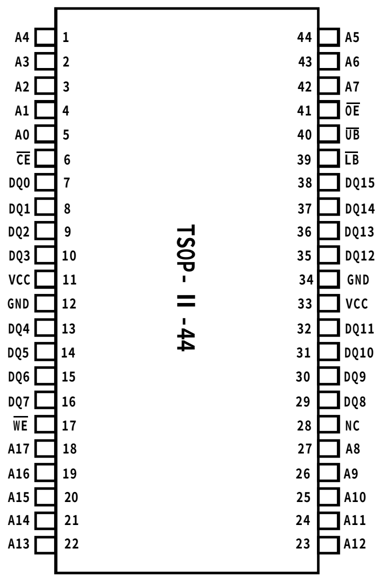
STC62WV1M8



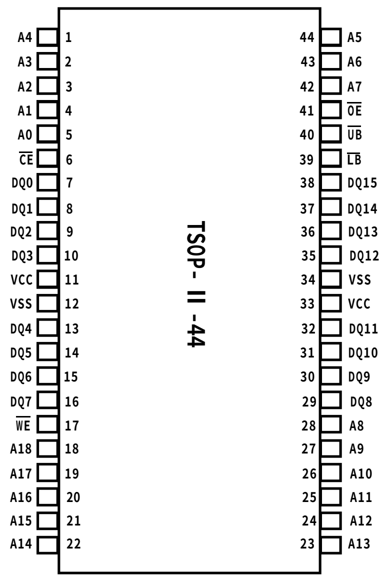
STC62WV12816



STC62WV25616



STC62WV51216



附录 I: STC 单片机配套工具价格

1.STC 单片机 ISP 下载编程工具，人民币 50 元(可申请样品)
2.STC 单片机 ISP 到 ISP 脱机下载工具，人民币 200 元(脱机就是量产时无须电脑,并可限制次数)
3.STC89 系列单片机专用烧录器，人民币 200 元(有些用户生产时希望擦除 ISP,或要烧录自己开发的 ISP 程序，用户自己开发的 ISP 程序可与用户应用程序一起烧录)

STC89C51RC/RD+ 系列 5V 单片机批量价格，1K 以下零售加 0.3 元，快递费 25 元

型 号	尾 缀	封 装	商规/工规	单 价	RAM 字节	Flash 程序	P4 口
STC89C51RC	40C-PDIP	DIP-40	商规	5.0元	512	4K	
STC89C51RC	40C-PLCC	PLCC-44	商规	5.2元	512	4K	有
STC89C51RC	40C-PQFP	PQFP-44	商规	5.6元	512	4K	有
STC89C51RC	40I-PDIP	DIP-40	工规	5.5元	512	4K	
STC89C51RC	40I-PLCC	PLCC-44	工规	5.7元	512	4K	有
STC89C51RC	40I-PQFP	PQFP-44	工规	6.0元	512	4K	有
STC89C52RC	40C-PDIP	DIP-40	商规	5.4元	512	8K	
STC89C52RC	40C-PLCC	PLCC-44	商规	6.0元	512	8K	有
STC89C52RC	40C-PQFP	PQFP-44	商规	6.0元	512	8K	有
STC89C52RC	40I-PDIP	DIP-40	工规	6.7元	512	8K	
STC89C52RC	40I-PLCC	PLCC-44	工规	6.7元	512	8K	有
STC89C52RC	40I-PQFP	PQFP-44	工规	7.0元	512	8K	有
STC89C53RC	40C-PQFP	DIP-40	商规	8.0元	512	15K	
STC89C53RC	40C-PLCC	PLCC-44	商规	8.0元	512	15K	有
STC89C53RC	40C-PQFP	PQFP-44	商规	8.0元	512	15K	有
STC89C53RC	40I-PQFP	DIP-40	工规	9.0元	512	15K	
STC89C53RC	40I-PLCC	PLCC-44	工规	9.0元	512	15K	有
STC89C53RC	40I-PQFP	PQFP-44	工规	9.0元	512	15K	有
STC89C54RD+	40C-PQFP	DIP-40	商规	9.0元	1280	16K	
STC89C54RD+	40C-PLCC	PLCC-44	商规	9.0元	1280	16K	有
STC89C54RD+	40C-PQFP	PQFP-44	商规	9.0元	1280	16K	有
STC89C54RD+	40I-PQFP	DIP-40	工规	11元	1280	16K	
STC89C54RD+	40I-PLCC	PLCC-44	工规	11元	1280	16K	有
STC89C54RD+	40I-PQFP	PQFP-44	工规	11元	1280	16K	有
STC89C58RD+	40C-PQFP	DIP-40	商规	12元	1280	32K	
STC89C58RD+	40C-PLCC	PLCC-44	商规	12元	1280	32K	有
STC89C58RD+	40C-PQFP	PQFP-44	商规	12元	1280	32K	有
STC89C58RD+	40I-PDIP	DIP-40	工规	15元	1280	32K	
STC89C58RD+	40I-PLCC	PLCC-44	工规	15元	1280	32K	有
STC89C58RD+	40I-PQFP	PQFP-44	工规	15元	1280	32K	有
STC89C516RD+	40C-PDIP	DIP-40	商规	13.5元	1280	63K	
STC89C516RD+	40C-PLCC	PLCC-44	商规	13.5元	1280	63K	有
STC89C516RD+	40C-PQFP	PQFP-44	商规	13.5元	1280	63K	有
STC89C516RD+	40I-PDIP	DIP-40	工规	17元	1280	63K	
STC89C516RD+	40I-PLCC	PLCC-44	工规	17元	1280	63K	有
STC89C516RD+	40I-PQFP	PQFP-44	工规	17元	1280	63K	有
STC89LE516AD				14元			
STC89LE58AD				12.5元			
STC89LE54AD				11元			
STC89LE52AD				9元			

附录 J: STC12C5410AD 系列 1T 单片机简介

---- 请从宏晶科技网站直接下载完整的 STC12C5410AD 系列单片机用户手册

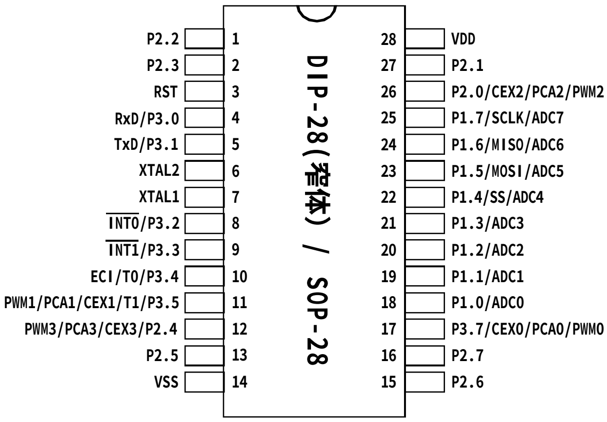
STC12C5410AD 系列单片机是宏晶科技推出的单时钟/机器周期(1T)的单片机，是高速/低功耗/超强抗干扰的新一代 8051 单片机，指令代码完全兼容传统 8051,但速度快 8-12 倍，内部集成 MAX810 专用复位电路。4 路 PWM,8 路高速 10 位 A/D 转换。

特 点 ：

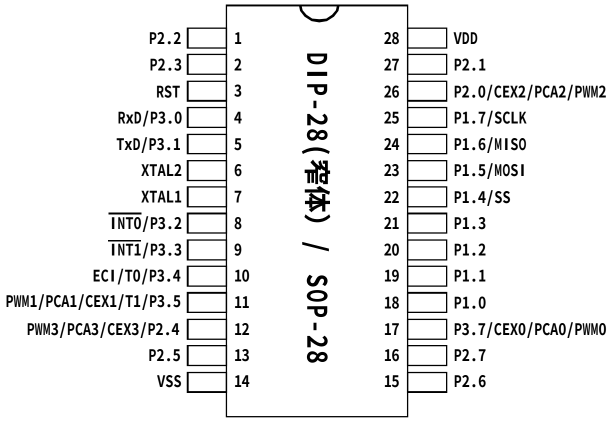
1. 增强型 8051 CPU，1T，单时钟/机器周期，指令代码完全兼容传统 8051
2. 工作电压：5.5V - 3.5V（5V 单片机） / 3.8V - 2.2V（3.3V 单片机）
3. 工作频率范围：0 - 35 MHz，相当于普通 8051 的 0～420MHz
4. 用户应用程序空间 12K / 10K / 8K / 6K / 4K / 2K / 1K 字节
5. 片上集成 512 字节 RAM
6. 通用 I/O 口（27/23 个），复位后为： 准双向口/弱上拉（普通 8051 传统 I/O 口）
   可设置成四种模式：准双向口/弱上拉，推挽/强上拉，仅为输入/高阻，开漏
   每个 I/O 口驱动能力均可达到 20mA，但整个芯片最大不得超过 55mA
7. ISP（在系统可编程）/IAP（在应用可编程），无需专用编程器，无需专用仿真器
   可通过串口（P3.0/P3.1）直接下载用户程序，数秒即可完成一片
8. EEPROM 功能
9. 看门狗
10. 内部集成 MAX810 专用复位电路（外部晶体 20M 以下时，可省外部复位电路）
11. 时钟源：外部高精度晶体/时钟，内部 R/C 振荡器
    用户在下载用户程序时，可选择是使用内部 R/C 振荡器还是外部晶体/时钟
    常温下内部 R/C 振荡器频率为：5.2MHz ～ 6.8MHz
    精度要求不高时，可选择使用内部时钟，但因为有制造误差和温漂，应认为是 4MHz ～ 8MHz
12. 共 2 个 16 位定时器/计数器，但可用 PCA 模块再产生 4 个定时器
13. 外部中断 2 路，下降沿中断或低电平触发中断，Power Down 模式可由外部中断低电平触发中断方式唤醒
14. PWM（4 路）/PCA（可编程计数器阵列）
    --- 也可用来当 4 个 D/A 使用
    --- 也可用来再实现 4 个定时器
    --- 也可用来再实现 4 个外部中断(上升沿中断/下降沿中断均可分别或同时支持)
15. ADC，10 位精度 ADC，共 8 路
16. 通用全双工异步串行口(UART)，由于 STC12 系列是高速的 8051，也可再用定时器软件实现多串口
17. SPI 同步通信口，主模式/从模式
18. 工作温度范围： 0 - 75°C / -40 - +85°C
19. 封装：SOP-32/28/20,LQFP-32，SKDIP-28，PDIP-20，TSSOP-20(超小封状 6.4mm × 6.4mm，定货)
    SOP-32 有 27 个 I/O 口，SOP28/SKDIP28 有 23 个 I/O 口，SOP20/TSSOP20/PDIP20 有 15 个 I/O 口，
    I/O 口不够时，可用 74HC165/74HC164/74HC595 串行扩展 I/O 口，或用双 CPU,三线通信，还多了串口。

STC12C5410AD 系列 1T 单片机管脚图

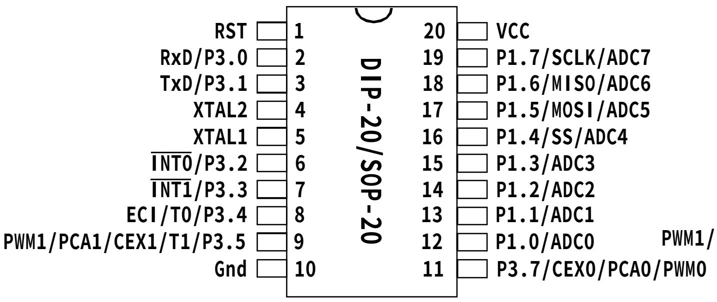
强烈推荐选择 SOP-20/28/32 贴片封装,尽量不选落后的插件 DIP 封装



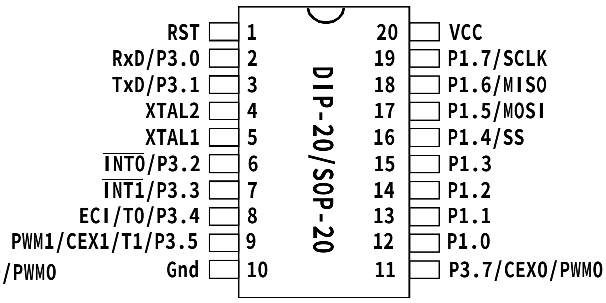
STC12C5410AD（有 A/D 转换）



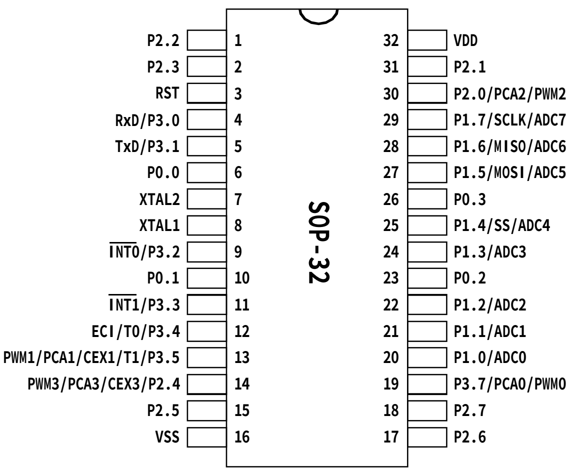
STC12C5410（无 A/D 转换）

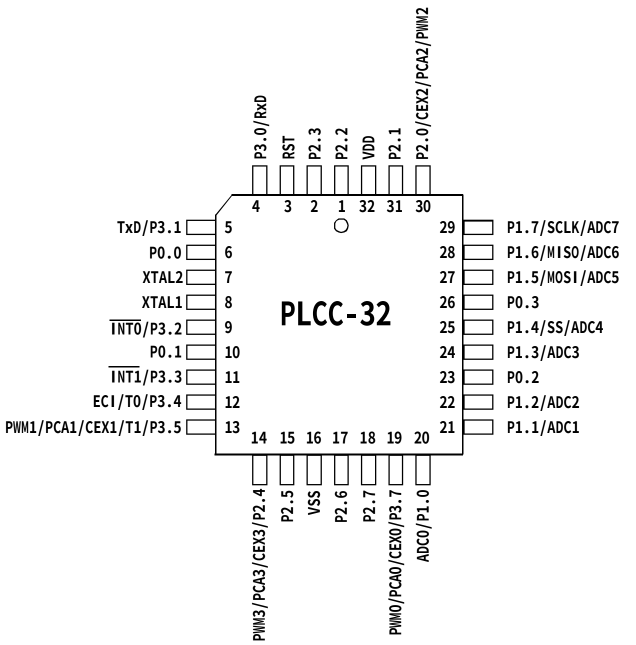


STC12C5410AD（有 A/D 转换）



STC12C5410（无 A/D 转换）



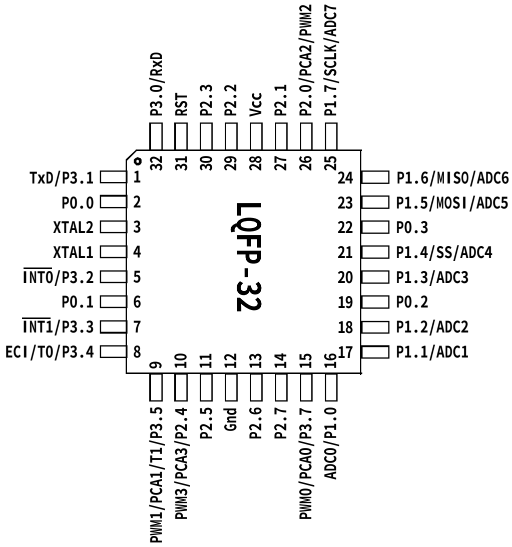


如选 32-Pin，推荐选 SOP-32，PLCC-32 不再生产

STC12C5410AD 系列（有 A/D 转换），32-Pin，有 27 个 I/O 口

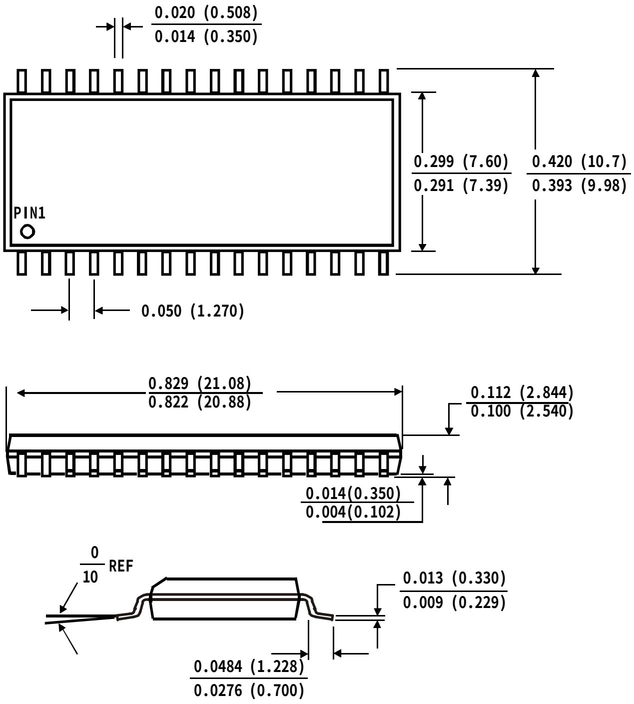
如须特别满足更高层次的Green 标准，请采用LQFP-32 封装

长 x 宽 = 9mm x 9mm, 高 < 1.6mm

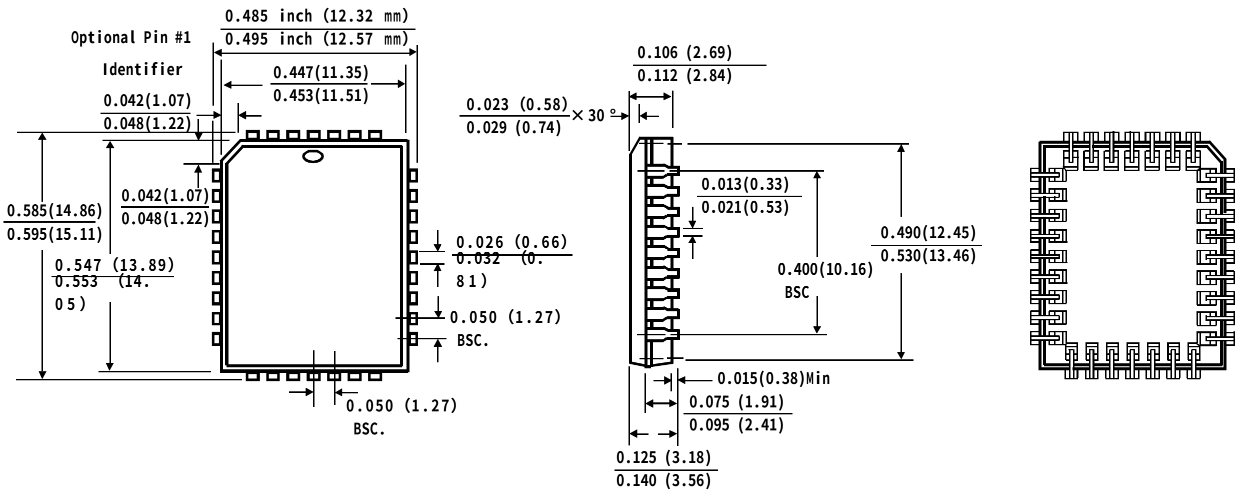


STC12C5410AD 系列 1T 单片机封装尺寸图

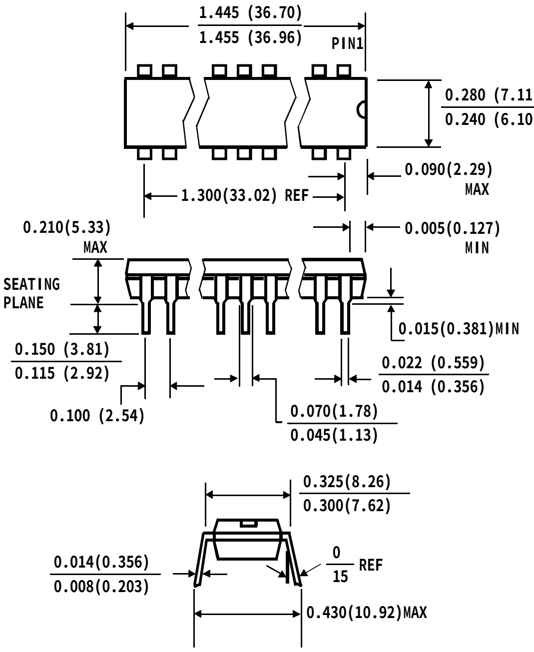
32-PIN SMALL OUTLINE PACKAGE (SOP-32)



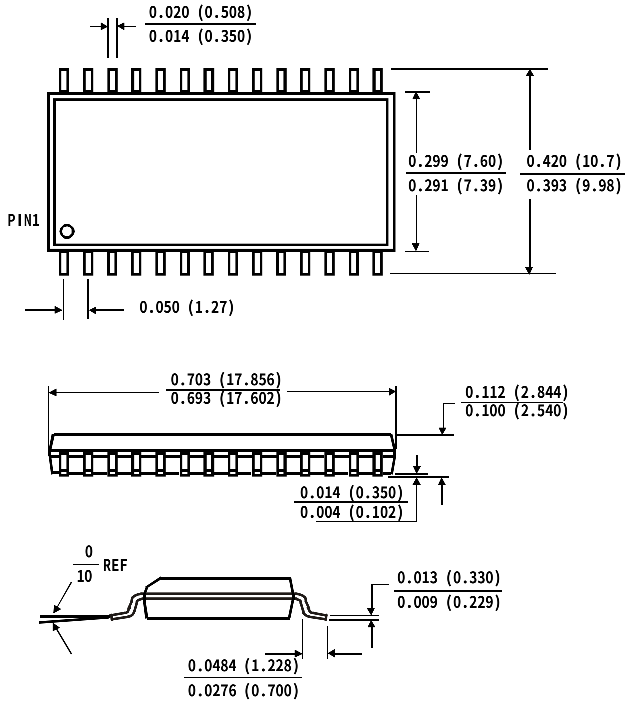
32-PIN PLASTIC LEAD CHIP CARRIER (PLCC)



28-PIN PLASTIC DUAL-IN-LINE PACKAGE (PDIP)



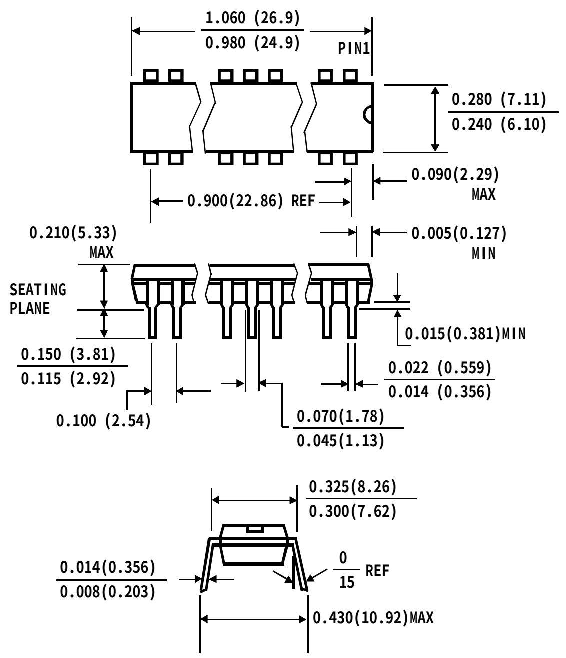
28-PIN SMALL OUTLINE PACKAGE (SOP)



20P3, 20-lead, 0.300" Wide, Plastic Dual Inline Package (PDIP-20)

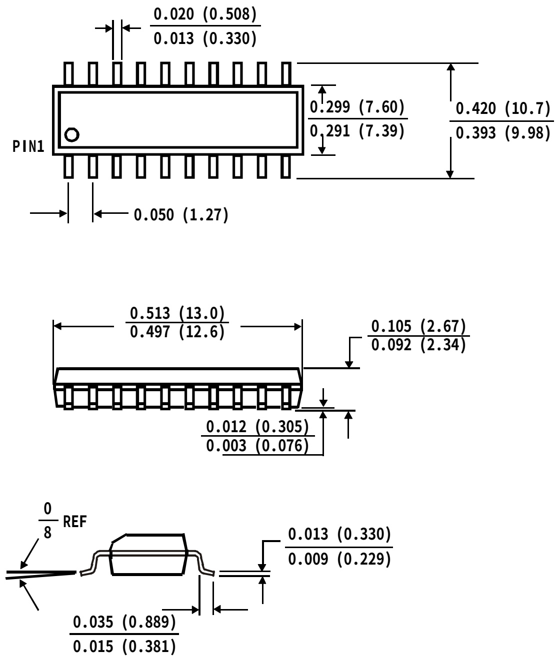
Dimensions in Inches and (Millimeters)

JEDEC STANDARD MS-001 AD



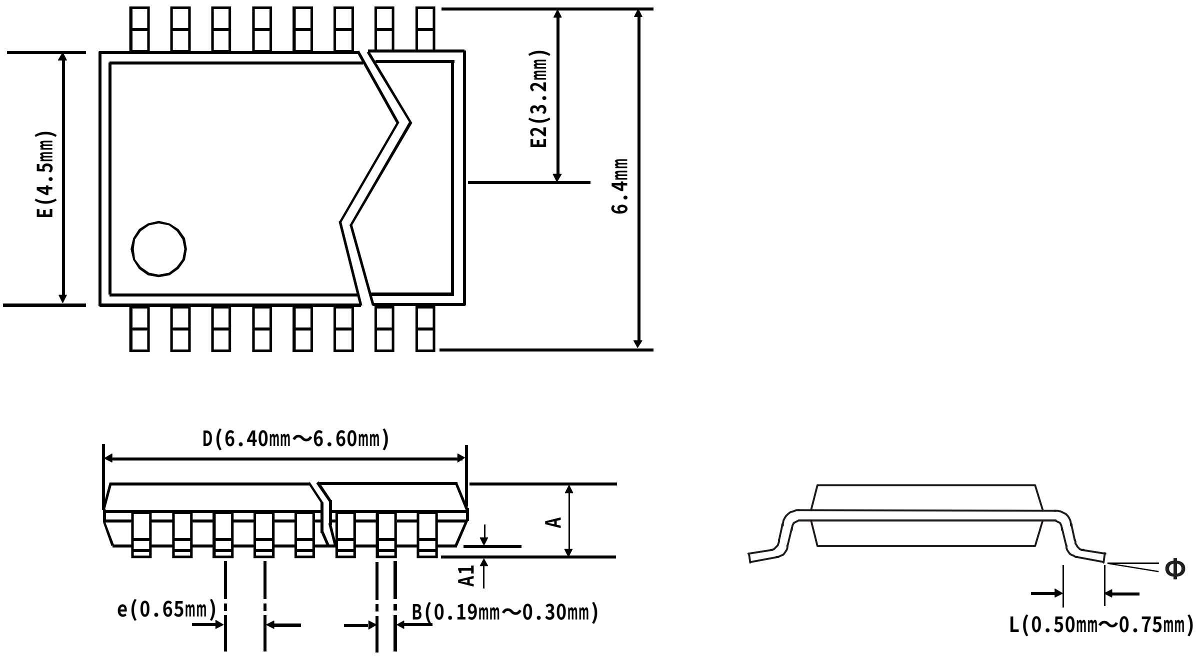
20S, 20-lead, 0.300" Wide, Plastic Gull WIng Small Outline (SOIC-20 / SOP-20)

Dimensions in Inches and (Millimeters)



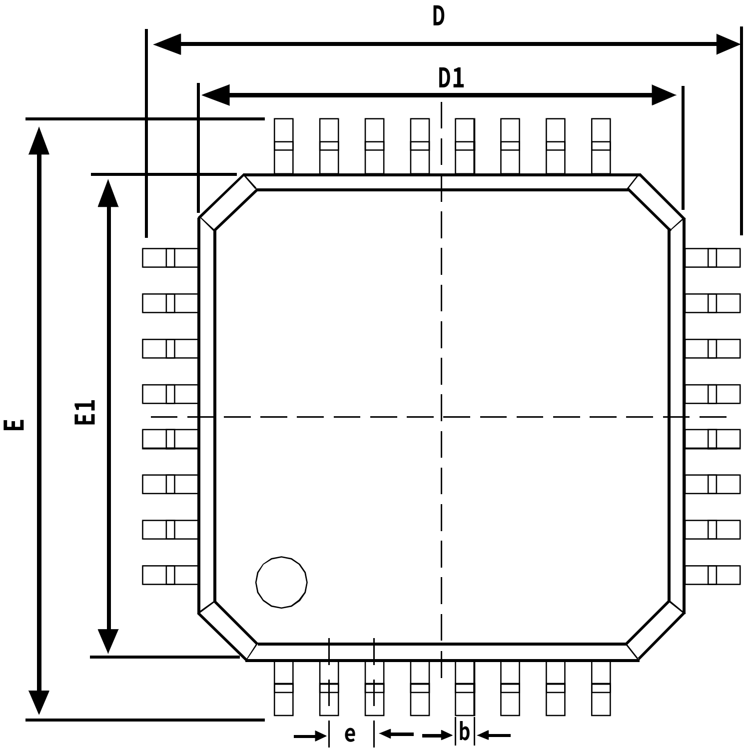
STC12C5410AD系列的TSSOP-20封装需订货

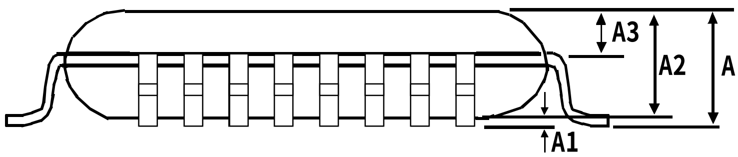
PACKAGE：PLASTIC SHRINK SMALL OUTLINE (TSSOP-20, 6.4mm × 6.4mm)

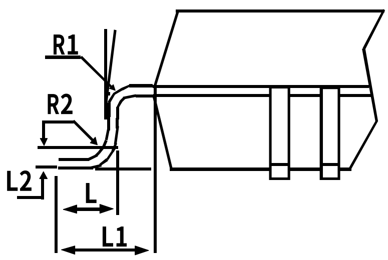


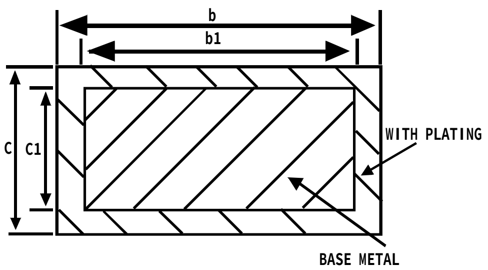
DIMENSIONS in inches (mm) Minimum/Maximum	20-PIN
A	- /0.043 (- /1.10)
A1	0.002/0.006 (0.05/0.15mm)
B	0.007/0.012 (0.19/0.30mm)
D	0.252/0.260 (6.40/6.60mm)
E	0.169/0.177 (4.30/4.50mm)
e	0.026 BSC (0.65mm BSC)
E2	0.126 BSC (3.20mm BSC)
L	0.020/0.030 (0.50/0.75mm)
Φ	0°/8°

LQFP-32 OUTLINE PACKAGE









COMMON DIMENSIONS			
(UNITS OF MEASURE = MILLIMETER)			
SYMBOL	MIN	NOM	MAX
A	-	-	1.60
A1	0.05	-	0.15
A2	1.35	1.40	1.45
A3	0.59	0.64	0.69
b	0.32	-	0.43
b1	0.31	0.35	0.39
c	0.13	-	0.18
c1	0.12	0.127	0.134
D	8.80	9.00	9.20
D1	6.90	7.00	7.10
E	8.80	9.00	9.20
E1	6.90	7.00	7.10
e	0.80BSC		
L	0.45	0.60	0.75
L1	1.00REF		
L2	0.25BSC		
R1	0.08	-	-
R2	0.08	-	0.20
S	0.20	-	-
Φ	0°	3.5°	7°
Φ1	0°	–	–
Φ2	11°	12°	13°
Φ3	11°	12°	13°

STC12C5410AD系列单片机选型一览表

型 号	工作电压(V)	Flash程序存储器字节	SRAM字节	定时器	UART	PCA 16位PWM 8位	A/D 8路	I/O	看门狗	内置复位	EEPROM	SPI	封装20-Pin	封装28-Pin	封装32-Pin
STC12C2052AD系列单片机选型一览															
STC12C1052	5.5 - 3.5	1K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP	管脚兼容 89C2051 超强抗干扰 无法解密	
STC12C1052AD	5.5 - 3.5	1K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12C2052	5.5 - 3.5	2K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP		
STC12C2052AD	5.5 - 3.5	2K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12C4052	5.5 - 3.5	4K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP		
STC12C4052AD	5.5 - 3.5	4K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12C5052	5.5 - 3.5	5K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP		
STC12C5052AD	5.5 - 3.5	5K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE1052	2.2 - 3.8	1K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE1052AD	2.2 - 3.8	1K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE2052	2.2 - 3.8	2K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE2052AD	2.2 - 3.8	2K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE4052	2.2 - 3.8	4K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE4052AD	2.2 - 3.8	4K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE5052	2.2 - 3.8	5K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE5052AD	2.2 - 3.8	5K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12C5410AD系列单片机选型一览															
STC12C5402	5.5 - 3.5	2K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5402AD	5.5 - 3.5	2K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5404	5.5 - 3.5	4K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5404AD	5.5 - 3.5	4K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5406	5.5 - 3.5	6K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5406AD	5.5 - 3.5	6K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5408	5.5 - 3.5	8K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5408AD	5.5 - 3.5	8K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5410	5.5 - 3.5	10K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5410AD	5.5 - 3.5	10K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5412	5.5 - 3.5	12K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5412AD	5.5 - 3.5	12K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5402	2.2 - 3.8	2K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5402AD	2.2 - 3.8	2K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5404	2.2 - 3.8	4K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5404AD	2.2 - 3.8	4K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5406	2.2 - 3.8	6K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5406AD	2.2 - 3.8	6K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5408	2.2 - 3.8	8K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5408AD	2.2 - 3.8	8K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5410	2.2 - 3.8	10K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5410AD	2.2 - 3.8	10K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5412	2.2 - 3.8	12K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5412AD	2.2 - 3.8	12K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP

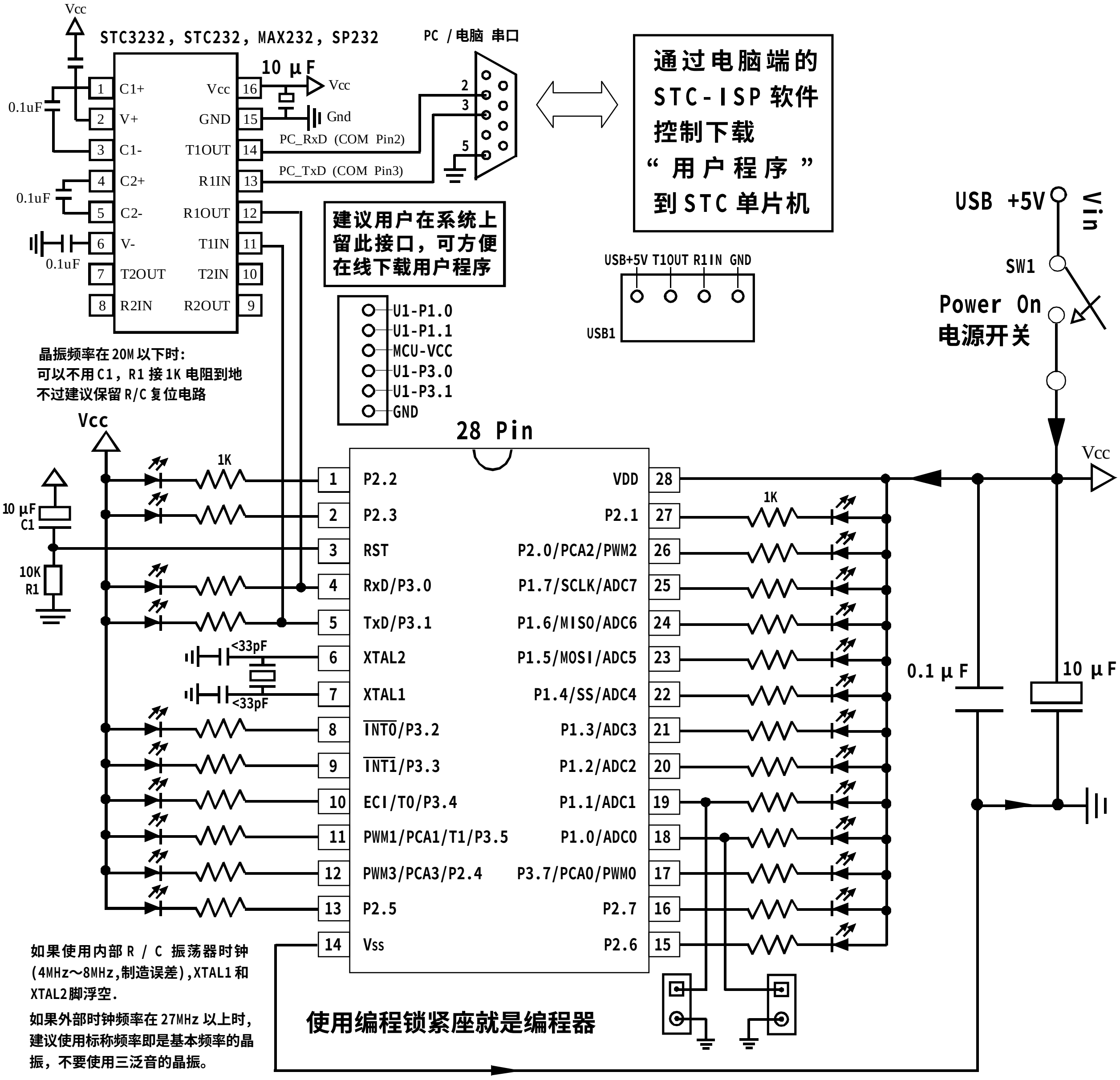
STC12C5410AD 系列 单片机 命名规则

STC12xx54xx xx — 35x - xxxx xx

- xx（最后）：管脚数 如 20，28，32
- xxxx：何种封装：如 PDIP，SOP，PLCC，TSSOP
- x：工作温度范围：
  I：工业级，-40℃ ～ +85℃
  C：商业级，0℃ ～ 70℃
- 35：工作频率：
  12：工作频率可到 12MHz
  35：工作频率可到 35MHz
  48：工作频率可到 48MHz
- xx：AD：有 A/D 转换功能
  无 AD 字样，则无 A/D 转换
- xx（54 后）：程序空间大小，如：
  04 是 4K 字节， 08 是 8K 字节
  10 是 10K 字节，12 是 12K 字节
- 54：RAM 是 512 字节，PCA/PWM 是 4 路
- xx（12 后）：
  C： 5.5V～3.5V
  LE：2.2V～3.8V
  LV：2.2V～3.8V,超低功耗设计,工作频率 2MHz 时:
  正常工作电流 < 2mA;
  空闲模式电流 < 1mA;
  掉电模式电流 < 0.1 μA
- 12：STC 1T 8051,同样的工作频率时,速度是普通 8051 的 8～12 倍

STC12C5410AD 系列 单片机 典型应用电路(28Pin)

---- 通过 RS-232 转换器连接电脑就可以下载程序



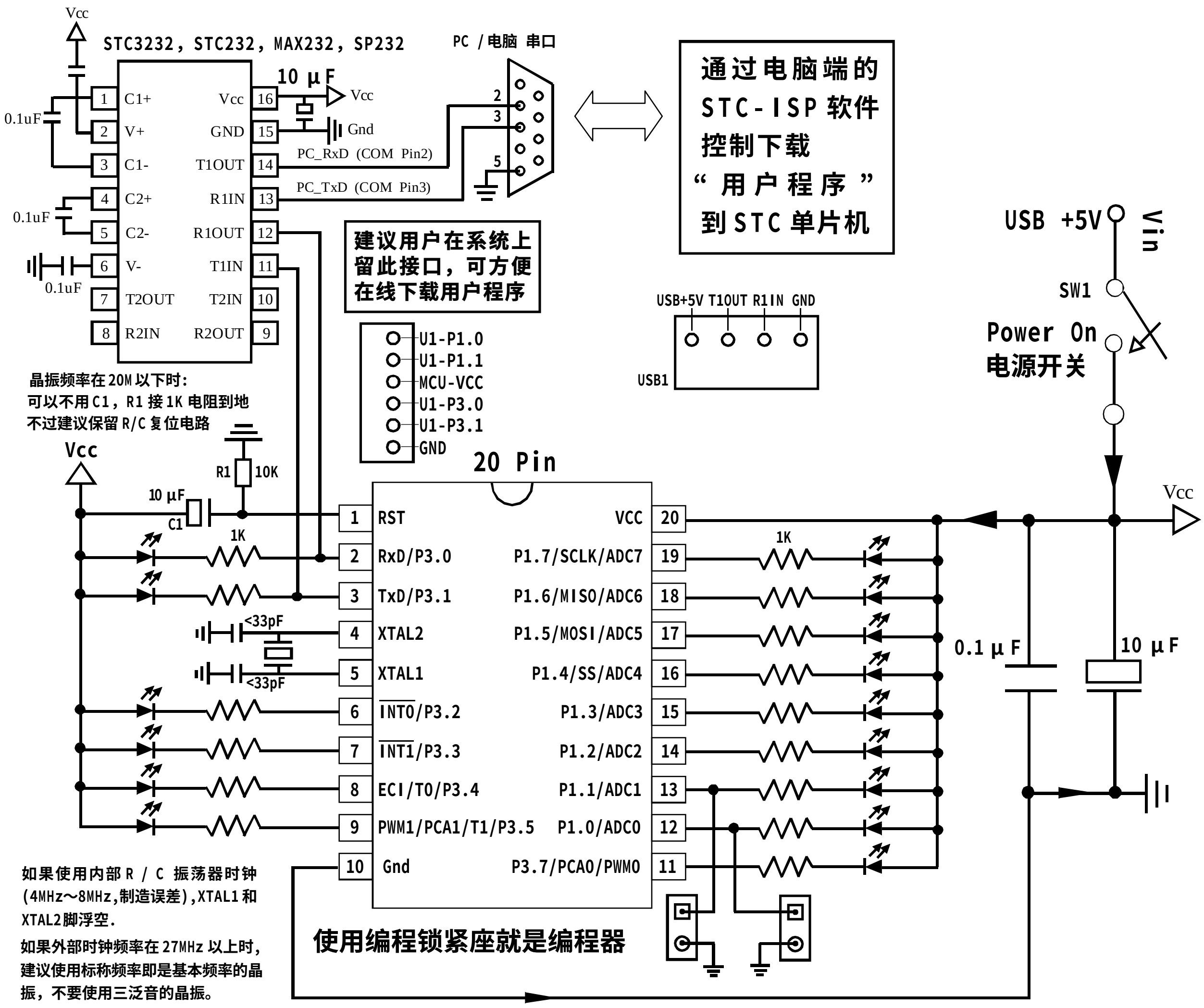
此线路已做成一个 STC12C5410AD 系列单片机　ISP 下载编程工具，可直接赠送给客户

用户在自己的目标系统上，如将 P3.0/P3.1 经过 RS-232 电平转换器转换后连接到电脑的普通 RS-232 串口，就可以在系统编程 / 升级用户软件。建议如果用户板上无 RS-232 电平转换器，应引出一个插座，含 Gnd / P3.1 / P3.0 / Vcc 四个信号线，当然如能引出 Gnd / P3.1 / P3.0 / Vcc / P1.1 / P1.0 六个信号线为最好，这样就可以在用户系统上直接编程了。关于 ISP 编程的原理及应用指南详见附录部分“STC12C5410AD 系列单片机 ISP 编程 原理 工具使用说明”部分。另外我们有标准化的编程下载工具，用户可以在上面编程后再插到目标系统上，也可以借用它上面的 RS-232 电平转换器连接到电脑，以做下载编程之用。编程一个芯片大致需几秒钟，速度比普通的通用编程器快很多，故无须买通用编程器。

电脑端 STC-ISP 软件从网站 www.MCU-Memory.com 下载

STC12C5410AD 系列 单片机 典型应用电路(20Pin)

---- 通过 RS-232 转换器连接电脑就可以下载程序



此线路已做成一个 STC12C5410AD 系列单片机 ISP 下载编程工具，可直接赠送给客户

用户在自己的目标系统上，如将 P3.0/P3.1 经过 RS-232 电平转换器转换后连接到电脑的普通 RS-232 串口，就可以在系统编程 / 升级用户软件。建议如果用户板上无 RS-232 电平转换器，应引出一个插座，含 Gnd / P3.1 / P3.0 / Vcc 四个信号线，当然如能引出 Gnd / P3.1 / P3.0 / Vcc / P1.1 / P1.0 六个信号线为最好，这样就可以在用户系统上直接编程了。关于 ISP 编程的原理及应用指南详见附录部分 " STC12C5410AD 系列单片机 ISP 编程 原理 工具使用说明 " 部分。另外我们有标准化的编程下载工具，用户可以在上面编程后再插到目标系统上，也可以借用它上面的 RS-232 电平转换器连接到电脑，以做下载编程之用。编程一个芯片大致需几秒钟，速度比普通的通用编程器快很多，故无须买通用编程器。

电脑端 STC-ISP 软件从网站 www.MCU-Memory.com 下载

指令系统分类总结及与普通 8051 指令执行时间对比

如果按功能分类，STC89/12 系列单片机指令系统可分为：
1.数据传送类指令；
2.算术操作类指令；
3.逻辑操作类指令；
4.控制转移类指令；
5.布尔变量操作类指令。
按功能分类的指令系统表如下表所示。

传统 12T 的 8051

STC12C5410AD 系列

数据传送类指令

助记符		功能说明	字节数	12时钟/机器周期所需时钟	1时钟/机器周期所需时钟	效率提升
MOV	A, Rn	寄存器内容送入累加器	1	12	1	12倍
MOV	A, direct	直接地址单元中的数据送入累加器	2	12	2	6倍
MOV	A, @Ri	间接RAM中的数据送入累加器	1	12	2	6倍
MOV	A, #data	立即送入累加器	2	12	2	6倍
MOV	Rn, A	累加器内容送入寄存器	1	12	2	6倍
MOV	Rn, direct	直接地址单元中的数据送入寄存器	2	24	4	6倍
MOV	Rn, #data	立即数送入寄存器	2	12	2	6倍
MOV	direct, A	累加器内容送入直接地址单元	2	12	3	4倍
MOV	direct, Rn	寄存器内容送入直接地址单元	2	24	3	8倍
MOV	direct, direct	直接地址单元中的数据送入另一个直接地址单元	3	24	4	6倍
MOV	direct, @Ri	间接RAM中的数据送入直接地址单元	2	24	4	6倍
MOV	direct, #data	立即数送入直接地址单元	3	24	3	8倍
MOV	@Ri, A	累加器内容送间接RAM单元	1	12	3	4倍
MOV	@Ri, direct	直接地址单元数据送入间接RAM单元	2	24	3	8倍
MOV	@Ri, #data	立即数送入间接RAM单元	2	12	3	4倍
MOV	DPTR, #data16	16位立即数送入地址寄存器	3	24	3	8倍
MOVC	A,@A+DPTR	以DPTR为基地址变址寻址单元中的数据送入累加器	1	24	4	6倍
MOVC	A,@A+PC	以PC为基地址变址寻址单元中的数据送入累加器	1	24	4	6倍
MOVX	A,@Ri	外部RAM（8位地址）送入累加器	1	24	3	8倍
MOVX	A,@DPTR	外部RAM（16位地址）送入累加器	1	24	3	8倍
MOVX	@Ri,A	累加器送外部RAM（8位地址）	1	24	3	8倍
MOVX	@DPTR ,A	累加器送外部RAM（16位地址）	1	24	3	8倍
PUSH	direct	直接地址单元中的数据压入堆栈	2	24	4	6倍
POP	direct	出栈送直接地址单元	2	24	3	8倍
XCH	A,Rn	寄存器与累加器交换	1	12	3	4倍
XCH	A,direct	直接地址单元与累加器交换	2	12	4	3倍
XCH	A,@Ri	间接RAM与累加器交换	1	12	4	3倍
XCHD	A,@Ri	间接RAM的低半字节与累加器交换	1	12	4	3倍

算术操作类指令

助记符		功能说明	字节数	12时钟/周期所需时钟	1时钟/周期所需时钟	提升效率
ADD	A, Rn	寄存器内容加到累加器	1	12	2	6倍
ADD	A, direct	直接地址单元中的数据加到累加器	2	12	3	4倍
ADD	A, @Ri	间接RAM中的数据加到累加器	1	12	3	4倍
ADD	A, #data	立即加到累加器	2	12	2	6倍
ADDC	A, Rn	寄存器内容带进位加到累加器	1	12	2	6倍
ADDC	A, direct	直接地址单元的内容带进位加到累加器	2	12	3	4倍
ADDC	A, @Ri	间接RAM内容带进位加到累加器	1	12	3	4倍
ADDC	A, #data	立即数带进位加到累加器	2	12	2	6倍
SUBB	A, Rn	累加器带借位减寄存器内容	1	12	2	6倍
SUBB	A, direct	累加器带借位减直接地址单元的内容	2	12	3	4倍
SUBB	A, @Ri	累加器带借位减间接RAM中的内容	1	12	3	4倍
SUBB	A, #data	累加器带借位减立即数	2	12	2	6倍
INC	A	累加器加1	1	12	2	6倍
INC	Rn	寄存器加1	1	12	3	4倍
INC	direct	直接地址单元加1	2	12	4	3倍
INC	@Ri	间接RAM单元加1	1	12	4	3倍
DEC	A	累加器减1	1	12	2	6倍
DEC	Rn	寄存器减1	1	12	3	4倍
DEC	direct	直接地址单元减1	2	12	4	3倍
DEC	@Ri	间接RAM单元减1	1	12	4	3倍
INC	DPTR	地址寄存器DPTR加1	1	24	1	24倍
MUL	AB	A乘以B	1	48	4	12倍
DIV	AB	A除以B	1	48	5	9.6倍
DA	A	累加器十进制调整	1	12	4	3倍

逻辑操作类指令

助记符		功能说明	字节数	12时钟/周期所需时钟	1时钟/周期所需时钟	提升效率
ANL	A, Rn	累加器与寄存器相“与”	1	12	2	6倍
ANL	A, direct	累加器与直接地址单元相“与”	2	12	3	4倍
ANL	A, @Ri	累加器与间接RAM单元相“与”	1	12	3	4倍
ANL	A, #data	累加器与立即数相“与”	2	12	2	6倍
ANL	direct, A	直接地址单元与累加器相“与”	2	12	4	3倍
ANL	direct, #data	直接地址单元与立即数相“与”	3	24	4	6倍
ORL	A, Rn	累加器与寄存器相“或”	1	12	2	6倍
ORL	A, direct	累加器与直接地址单元相“或”	2	12	3	4倍
ORL	A, @Ri	累加器与间接RAM单元相“或”	1	12	3	4倍
ORL	A, #data	累加器与立即数相“或”	2	12	2	6倍
ORL	direct, A	直接地址单元与累加器相“或”	2	12	4	3倍
ORL	direct, #data	直接地址单元与立即数相“或”	3	24	4	6倍
XRL	A, Rn	累加器与寄存器相“异或”	1	12	2	6倍
XRL	A, direct	累加器与直接地址单元相“异或”	2	12	3	4倍
XRL	A, @Ri	累加器与间接RAM单元相“异或”	1	12	3	4倍
XRL	A, #data	累加器与立即数相“异或”	2	12	2	6倍
XRL	direct, A	直接地址单元与累加器相“异或”	2	12	4	3倍
XRL	direct, #data	直接地址单元与立即数相“异或”	3	24	4	6倍
CLR	A	累加器清“0”	1	12	1	12倍
CPL	A	累加器求反	1	12	2	6倍
RL	A	累加器循环左移	1	12	1	12倍
RLC	A	累加器带进位位循环左移	1	12	1	12倍
RR	A	累加器循环右移	1	12	1	12倍
RRC	A	累加器带进位位循环右移	1	12	1	12倍
SWAP	A	累加器半字节交换	1	12	1	12倍

控制转移类指令

助记符		功能说明	字节数	12时钟/周期所需时钟	1时钟/周期所需时钟	提升效率
ACALL	addr11	绝对（短）调用子程序	2	24	6	4倍
LCALL	addr16	长调用子程序	3	24	6	4倍
RET		子程序返回	1	24	4	6倍
RETI		中断返回	1	24	4	6倍
AJMP	addr11	绝对（短）转移	2	24	3	8倍
LJMP	addr16	长转移	3	24	4	6倍
SJMP	rel	相对转移	2	24	3	8倍
JMP	@A+DPTR	相对于DPTR的间接转移	1	24	3	8倍
JZ	rel	累加器为零转移	2	24	3	8倍
JNZ	rel	累加器非零转移	2	24	3	8倍
CJNE	A，direct，rel	累加器与直接地址单元比较，不相等则转移	3	24	5	4.8倍
CJNE	A，#data，rel	累加器与立即数比较，不相等则转移	3	24	4	6倍
CJNE	Rn，#data，rel	寄存器与立即数比较，不相等则转移	3	24	4	6倍
CJNE	@Ri，#data，rel	间接RAM单元与立即数比较，不相等则转移	3	24	5	4.8倍
DJNZ	Rn，rel	寄存器减1，非零转移	3	24	4	6倍
DJNZ	direct，rel	直接地址单元减1，非零转移	3	24	5	4.8倍
NOP		空操作	1	12	1	12倍

布尔变量操作类指令

助记符		功能说明	字节数	12时钟/周期所需时钟	1时钟/周期所需时钟	提升效率
CLR	C	清0进位位	1	12	1	12倍
CLR	bit	清0直接地址位	2	12	4	3倍
SETB	C	置1进位位	1	12	1	12倍
SETB	bit	置1直接地址位	2	12	4	3倍
CPL	C	进位位求反	1	12	1	12倍
CPL	bit	直接地址位求反	2	12	4	3倍
ANL	C，bit	进位位和直接地址位相“与”	2	24	3	8倍
ANL	C，$\overline{\text{bit}}$	进位位和直接地址位的反码相“与”	2	24	3	8倍
ORL	C，bit	进位位和直接地址位相“或”	2	24	3	8倍
ORL	C，$\overline{\text{bit}}$	进位位和直接地址位的反码相“或”	2	24	3	8倍
MOV	C，bit	直接地址位送入进位位	2	12	3	4倍
MOV	bit，C	进位位送入直接地址位	2	24	3	8倍
JC	rel	进位位为1则转移	2	24	3	8倍
JNC	rel	进位位为0则转移	2	24	3	8倍
JB	bit，rel	直接地址位为1则转移	3	24	4	6倍
JNB	bit，rel	直接地址位为0则转移	3	24	4	6倍
JBC	bit，rel	直接地址位为1则转移，该位清0	3	24	5	4.8倍

指令执行速度效率提升总结：

指令系统共包括111 条指令，其中：

执行速度快24倍的	共1条
执行速度快12倍的	共12条
执行速度快9.6倍的	共1条
执行速度快8倍的	共21条
执行速度快6倍的	共37条
执行速度快4.8倍的	共4条
执行速度快4倍的	共21条
执行速度快3倍的	共14条

根据对指令的使用频率分析统计，STC12 系列 1T 的8051 单片机比普通的8051 单片机在同样的工作频率下运行速度提升了8～12 倍。

指令执行时钟数统计（供参考）：

指令系统共包括111 条指令，其中：

1个时钟就可执行完成的指令	共12条
2个时钟就可执行完成的指令	共20条
3个时钟就可执行完成的指令	共40条
4个时钟就可执行完成的指令	共32条
5个时钟就可执行完成的指令	共5条
6个时钟就可执行完成的指令	共2条

特殊功能寄存器映像　SFR Mapping

	Bit Addressable	Non Bit Addressable							
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	
F8h		CH 0000,0000	CCAP0H 0000,0000	CCAP1H 0000,0000	CCAP2H 0000,0000	CCAP3H 0000,0000			FFh
F0h	B 0000,0000		PCA_PWM0 xxxx,xx00	PCA_PWM1 xxxx,xx00	PCA_PWM2 xxxx,xx00	PCA_PWM3 xxxx,xx00			F7h
E8h		CL 0000,0000	CCAP0L 0000,0000	CCAP1L 0000,0000	CCAP2L 0000,0000	CCAP3L 0000,0000			EFh
E0h	ACC 0000,0000	WDT_CONTR 0x00,0000	ISP_DATA 1111,1111	ISP_ADDRH 0000,0000	ISP_ADDRL 0000,0000	ISP_CMD xxxx,xx00	ISP_TRIG xxxx,xxxx	ISP_CONTR 0000,1000	E7h
D8h	CCON 00xx,0000	CMOD 0xxx,x000	CCAPM0 x000,0000	CCAPM1 x000,0000	CCAPM2 x000,0000	CCAPM3 x000,0000			DFh
D0h	PSW 0000,0000								D7h
C8h									CFh
C0h						ADC_CONTR 0000,0000	ADC_DATA 0000,0000	CLK_DIV xxxx,x000	C7h
B8h	IP x000,0000	SADEN don't use					ADC_LOW2 0000,0000		BFh
B0h	P3 1x11,1111	P3M0 0000,0000	P3M1 0000,0000					IPH x000,0000	B7h
A8h	IE 0000,0000	SADDR don't use							AFh
A0h	P2 1111,1111							TEST_WDT don't use	A7h
98h	SCON 0000,0000	SBUF xxxx,xxxx							9Fh
90h	P1 1111,1111	P1M0 0000,0000	P1M1 0000,0000	P0M0 0000,0000	P0M1 0000,0000	P2M0 0000,0000	P2M1 0000,0000		97h
88h	TCON 0000,0000	TMOD 0000,0000	TL0 0000,0000	TL1 0000,0000	TH0 0000,0000	TH1 0000,0000	AUXR 0000,00xx		8Fh
80h	P0 xxxx,1111	SP 0000,0111	DPL 0000,0000	DPH 0000,0000	SPSTAT 00xx,xxxx	SPCTL 0000,0100	SPDAT 0000,0000	PCON 0011,0000	87h
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	

特别标出部分为在 Intel 8052 基础上新增加的特殊功能寄存器，一般用户可不管

STC12C5410AD 系列 8051 单片机内核特殊功能寄存器 C51 Core SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
ACC	E0h	Accumulator									0000,0000
B	F0h	B Register									0000,0000
PSW	D0h	Program Status Word	CY	AC	F0	RS1	RS0	OV	F1	P	0000,0000
SP	81h	Stack Pointer									0000,0111
DPL	82h	Data Pointer Low Byte									0000,0000
DPH	83h	Data Pointer High Byte									0000,0000

STC12C5410AD 系列 8051 单片机系统管理特殊功能寄存器 System Management SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
PCON	87h	Power Control	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	EADCI	ESPI	ELVDI	-	-	0000,00xx
CLK_DIV	C7h	Clock Divder	-	-	-	-	-	CLKS2	CLKS1	CLKS0	xxxx,x000

STC12C5410AD 系列 8051 单片机 I/O 口 特殊功能寄存器 Port SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P0	80h	8-bit Port 0	-	-	-	-	P0.3	P0.2	P0.1	P0.0	xxxx,1111
P0M0	93h										0000,0000
P0M1	94h										0000,0000
P1	90h	8-bit Port 1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	1111,1111
P1M0	91h										0000,0000
P1M1	92h										0000,0000
P2	A0h	8-bit Port 2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	1111,1111
P2M0	95h										0000,0000
P2M1	96h										0000,0000
P3	B0h	8-bit Port 3	P3.7	-	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	1x11,1111
P3M0	B1h										0000,0000
P3M1	B2h										0000,0000

STC12C5410AD 系列 8051 单片机 定时器 特殊功能寄存器 Timer SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
TCON	88h	Timer / Counter 0 and 1 Control	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000,0000
TMOD	89h	Timer / Counter 0 and 1 Modes	GATE GATE1	C/T# C/T1#	M1 M1_1	M0 M1_0	GATE GATE0	C/T# C/T0#	M1 M0_1	M0 M0_0	0000,0000
TL0	8Ah	Timer / Counter 0 Low Byte									0000,0000
TH0	8Ch	Timer / Counter 0 High Byte									0000,0000
TL1	8Bh	Timer / Counter 1 Low Byte									0000,0000
TH1	8Dh	Timer / Counter 1 High Byte									0000,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	EADCI	ESPI	ELVDI	-	-	0000,00xx

STC12C5410AD 系列 8051 单片机 串行口 特殊功能寄存器 Serial I/O Port SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
SCON	98h	Serial Control	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000,0000
SBUF	99h	Serial Data Buffer									xxxx,xxxx
SADEN	B9h	Slave Address Mask									0000,0000
SADDR	A9h	Slave Address									0000,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	EADCI	ESPI	ELVDI	-	-	0000,00xx

STC12C5410AD 系列 8051 单片机 看门狗定时器 特殊功能寄存器 Watch Dog Timer SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
WDT_CONTR	E1h	Watch-Dog-Timer Control register	WDT_FLAG	-	EN_WDT	CLR_WDT	IDLE_WDT	PS2	PS1	PS0	xx00,0000

STC12C5410AD 系列 1T 8051 单片机 中断 特殊功能寄存器 Interrupt SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
IE	A8h	Interrupt Enable	EA	EPCA_LVD	EADC_SPI	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	-	PPCA_LVD	PADC_SPI	PS	PT1	PX1	PT0	PX0	xx00,0000
IPH	B7h	Interrupt Priority High	-	PPCA_LVDH	PADC_SPIH	PSH	PT1H	PX1H	PT0H	PX0H	0000,0000
TCON	88h	Timer / Counter 0 and 1 Control	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000,0000
SCON	98h	Serial Control	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000,0000
PCON	87h	Power Control	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	EADCI	ESPI	ELVDI	-	-	0000,00xx
ADC_CONTR	C5h	A/D 转换控制寄存器	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	0xx0,0000
SPSTAT	84h	SPI Status Register	SPIF	WCOL	-	-	-	-	-	-	00xx,xxxx
CCON	D8h	PCA Control Register	CF	CR	-	-	CCF3	CCF2	CCF1	CCF0	00xx,0000
CMOD	D9h	PCA Mode Register	CIDL	-	-	-	-	CPS1	CPS0	ECF	0xxx,x000
CCAPM0	DAh	PCA Module 0 Mode Register	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x000,0000
CCAPM1	DBh	PCA Module 1 Mode Register	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x000,0000
CCAPM2	DCh	PCA Module 2 Mode Register	-	ECOM2	CAPP2	CAPN2	MAT2	TOG2	PWM2	ECCF2	x000,0000
CCAPM3	DDh	PCA Module 3 Mode Register	-	ECOM3	CAPP3	CAPN3	MAT3	TOG3	PWM3	ECCF3	x000,0000

STC12C5410AD 系列 8051 单片机 PCA/PWM 特殊功能寄存器 PCA/PWM SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
CCON	D8h	PCA Control Register	CF	CR	-	-	CCF3	CCF2	CCF1	CCF0	00xx,0000
CMOD	D9h	PCA Mode Register	CIDL	-	-	-	-	CPS1	CPS0	ECF	0xxx,x000
CCAPM0	DAh	PCA Module 0 Mode Register	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x000,0000
CCAPM1	DBh	PCA Module 1 Mode Register	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x000,0000
CCAPM2	DCh	PCA Module 2 Mode Register	-	ECOM2	CAPP2	CAPN2	MAT2	TOG2	PWM2	ECCF2	x000,0000
CCAPM3	DDh	PCA Module 3 Mode Register	-	ECOM3	CAPP3	CAPN3	MAT3	TOG3	PWM3	ECCF3	x000,0000
CL	E9h	PCA Base Timer Low									0000,0000
CH	F9h	PCA Base Timer High									0000,0000
CCAP0L	EAh	PCA Module-0 Capture Register Low									0000,0000
CCAP0H	FAh	PCA Module-0 Capture Register High									0000,0000
CCAP1L	EBh	PCA Module-1 Capture Register Low									0000,0000
CCAP1H	FBh	PCA Module-1 Capture Register High									0000,0000
CCAP2L	ECh	PCA Module-2 Capture Register Low									0000,0000
CCAP2H	FCh	PCA Module-2 Capture Register High									0000,0000
CCAP3L	EDh	PCA Module-3 Capture Register Low									0000,0000
CCAP3H	FDh	PCA Module-3 Capture Register High									0000,0000
PCA_PWM0	F2h	PCA PWM Mode Auxiliary Register 0	-	-	-	-	-	-	EPC0H	EPC0L	xxxx,xx00
PCA_PWM1	F3h	PCA PWM Mode Auxiliary Register 1	-	-	-	-	-	-	EPC1H	EPC1L	xxxx,xx00
PCA_PWM2	F4h	PCA PWM Mode Auxiliary Register 2	-	-	-	-	-	-	EPC2H	EPC2L	xxxx,xx00
PCA_PWM3	F5h	PCA PWM Mode Auxiliary Register 3	-	-	-	-	-	-	EPC3H	EPC3L	xxxx,xx00

STC12C5410AD系列 8051 单片机 ISP/IAP 特殊功能寄存器 ISP/IAP SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
ISP_DATA	E2h	ISP/IAP Flash Data Register									1111,1111
ISP_ADDRH	E3h	ISP/IAP Flash Address High									0000,0000
ISP_ADDRL	E4h	ISP/IAP Flash Address Low									0000,0000
ISP_CMD	E5h	ISP/IAP Flash Command Register	-	-	-	-	-	-	MS1	MS0	xxxx,x000
ISP_TRIG	E6h	ISP/IAP Flash Command Trigger									xxxx,xxxx
ISP_CONTR	E7h	ISP/IAP Control Register	ISPEN	SWBS	SWRST	CMD_FAIL	1	WT2	WT1	WT0	0000,1000

中断

STC12C5410AD 系列 1T 8051 单片机 中断 特殊功能寄存器 Interrupt SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
IE	A8h	Interrupt Enable	EA	EPCA_LVD	EADC_SPI	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	-	PPCA_LVD	PADC_SPI	PS	PT1	PX1	PT0	PX0	xx00,0000
IPH	B7h	Interrupt Priority High	-	PPCA_LVDH	PADC_SPIH	PSH	PT1H	PX1H	PT0H	PX0H	0000,0000
TCON	88h	Timer / Counter 0 and 1 Control	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000,0000
SCON	98h	Serial Control	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000,0000
PCON	87h	Power Control	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	EADCI	ESPI	ELVDI	-	-	0000,00xx
ADC_CONTR	C5h	A/D 转换控制寄存器	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	0xx0,0000
SPSTAT	84h	SPI Status Register	SPIF	WCOL	-	-	-	-	-	-	00xx,xxxx
CCON	D8h	PCA Control Register	CF	CR	-	-	CCF3	CCF2	CCF1	CCF0	00xx,0000
CMOD	D9h	PCA Mode Register	CIDL	-	-	-	-	CPS1	CPS0	ECF	0xxx,x000
CCAPM0	DAh	PCA Module 0 Mode Register	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x000,0000
CCAPM1	DBh	PCA Module 1 Mode Register	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x000,0000
CCAPM2	DCh	PCA Module 2 Mode Register	-	ECOM2	CAPP2	CAPN2	MAT2	TOG2	PWM2	ECCF2	x000,0000
CCAPM3	DDh	PCA Module 3 Mode Register	-	ECOM3	CAPP3	CAPN3	MAT3	TOG3	PWM3	ECCF3	x000,0000

STC12C5410AD 系列 中断与普通 8051 完全兼容，优先级可设为 4 级。

Interrupt Source 中断源	Vector Address 中断向量地址	Polling Sequence 中断查询次序	中断优先级设置	优先级0 最低	优先级1	优先级2	优先级3 最高	Interrupt Request 中断请求
/INT0	0003H	0(最优先)	PX0H,PX0	0,0	0,1	1,0	1,1	IE0
Timer 0	000BH	1	PT0H,PT0	0,0	0,1	1,0	1,1	TF0
/INT1	0013H	2	PX1H,PX1	0,0	0,1	1,0	1,1	IE1
Timer 1	001BH	3	PT1H,PT1	0,0	0,1	1,0	1,1	IF1
UART	0023H	4	PSH, PS	0,0	0,1	1,0	1,1	RI + TI
ADC/SPI	002BH	5	PADC_SPIH,PADC_SPI	0,0	0,1	1,0	1,1	ADC_FLAG + SPIF
PCA/LVD	0033H	6	PPCA_LVDH,PPCA_LVD	0,0	0,1	1,0	1,1	CF + CCF0 + CCF1 + CCF2 + CCF3 + LVDF

PCA/PWM 特殊功能寄存器 ， 其中部分位与 PCA 中断有关

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
CCON	D8h	PCA Control Register	CF	CR	-	-	CCF3	CCF2	CCF1	CCF0	00xx,0000
CMOD	D9h	PCA Mode Register	CIDL	-	-	-	-	CPS1	CPS0	ECF	0xxx,x000
CCAPM0	DAh	PCA Module 0 Mode Register	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x000,0000
CCAPM1	DBh	PCA Module 1 Mode Register	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x000,0000
CCAPM2	DCh	PCA Module 2 Mode Register	-	ECOM2	CAPP2	CAPN2	MAT2	TOG2	PWM2	ECCF2	x000,0000
CCAPM3	DDh	PCA Module 3 Mode Register	-	ECOM3	CAPP3	CAPN3	MAT3	TOG3	PWM3	ECCF3	x000,0000
CL	E9h	PCA Base Timer Low									0000,0000
CH	F9h	PCA Base Timer High									0000,0000
CCAP0L	EAh	PCA Module-0 Capture Register Low									0000,0000
CCAP0H	FAh	PCA Module-0 Capture Register High									0000,0000
CCAP1L	EBh	PCA Module-1 Capture Register Low									0000,0000
CCAP1H	FBh	PCA Module-1 Capture Register High									0000,0000
CCAP2L	ECh	PCA Module-2 Capture Register Low									0000,0000
CCAP2H	FCh	PCA Module-2 Capture Register High									0000,0000
CCAP3L	EDh	PCA Module-3 Capture Register Low									0000,0000
CCAP3H	FDh	PCA Module-3 Capture Register High									0000,0000
PCA_PWM0	F2h	PCA PWM Mode Auxiliary Register 0	-	-	-	-	-	-	EPC0H	EPC0L	xxxx,xx00
PCA_PWM1	F3h	PCA PWM Mode Auxiliary Register 1	-	-	-	-	-	-	EPC1H	EPC1L	xxxx,xx00
PCA_PWM2	F4h	PCA PWM Mode Auxiliary Register 2	-	-	-	-	-	-	EPC2H	EPC2L	xxxx,xx00
PCA_PWM3	F5h	PCA PWM Mode Auxiliary Register 3	-	-	-	-	-	-	EPC3H	EPC3L	xxxx,xx00

STC12C5410AD 系列 8051 单片机 SPI 功能模块特殊功能寄存器　其中 SPIF 位与中断有关

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
SPCTL	85h	SPI Control Register	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	0000,0000
SPSTAT	84h	SPI Status Register	SPIF	WCOL	-	-	-	-	-	-	00xx,xxxx
SPDAT	86h	SPI Data Register									0000,0000

定时器0/ 定时器1，UART 串口的速度

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	EADCI	ESPI	ELVDI	-	-	0000,00xx

定时器0和定时器1：
STC12C5410AD 系列是 1T 的8051单片机，为了兼容传统8051，定时器0和定时器1复位后是传统8051的速度，即12 分频，这是为了兼容传统8051。但也可不进行12 分频，实现真正的1T。
T0x12: 0， 定时器0是传统8051速度，12分频；1， 定时器0的速度是传统8051的12倍，不分频
T1x12: 0， 定时器1是传统8051速度，12分频；1， 定时器1的速度是传统8051的12倍，不分频
UART串口的模式0:
STC12C2052AD系列是 1T 的8051单片机，为了兼容传统8051，UART串口复位后是兼容传统8051的。
UART_M0x6: 0， UART串口的模式0是传统12T的8051速度，12分频；
1，UART串口的模式0的速度是传统12T的8051的6倍，2分频

EADCI: 0，禁止A/D 中断； 1，允许A/D中断
ESPI: 0，禁止SPI 中断； 1，允许SPI中断
ELVDI: 0，禁止低压中断； 1，允许低压中断。
5V 单片机，3.7V以下为低压，3V 单片机，2.4V以下为低压，
如ELVDI=1 (允许低压中断)，则会产生低压中断

系统工作时钟

STC12C5410AD 系列是 1T 的 8051 单片机，系统时钟兼容传统 8051。

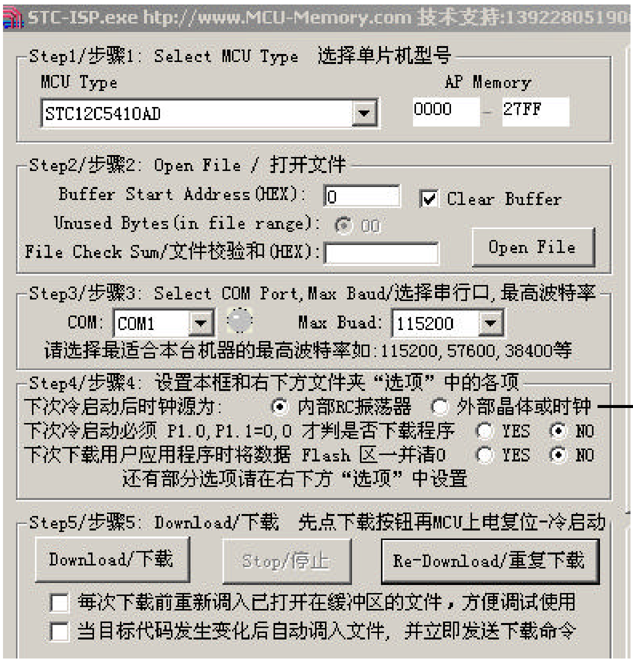
现出厂标准配置是使用芯片内部的 R/C 振荡器，5V 单片机常温下频率是 5.65MHz - 5.95MHz,因为随着温度的变化，内部 R/C 振荡器的频率会有一些温飘，故内部 R/C 振荡器只适用于对时钟频率要求不敏感的场合。

在对 STC12C5410AD 系列单片机进行 ISP 下载用户程序时，可以在选项中选择：

“ 下次冷启动后时钟源为外部晶体或时钟 ”

这样下载完用户程序后，停电，再冷启动后单片机的工作时钟使用的就不是内部 R/C 振荡器，而是外部晶体振荡后产生的高精度时钟了（接在 XTAL1/XTAL2 管脚上），也可以直接从 XTAL1 脚输入外部时钟，XTAL2 脚浮空。

如果还要设置成使用内部 R/C 振荡器，在对 STC12C5410AD 系列单片机进行 ISP 下载用户程序时，可以在选项中选择： “ 下次冷启动后时钟源为内部 R/C 振荡器 ”



系统时钟分频寄存器

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
CLK_DIV	C7h	Clock Divder	-	-	-	-	-	CLKS2	CLKS1	CLKS0	xxxx,x000

如用户系统希望大幅降低功耗，还可对系统时钟进行分频。（STC12C2052AD 系列不要使用分频）

CLKS2	CLKS1	CLKS0	CPU的实际工作时钟
0	0	0	系统时钟(外部时钟或内部R/C振荡时钟)
0	0	1	系统时钟/2
0	1	0	系统时钟/4
0	1	1	系统时钟/8
1	0	0	系统时钟/16
1	0	1	系统时钟/32
1	1	0	系统时钟/64
1	1	1	系统时钟/128

STC12C5410AD 系列单片机可以在空闲模式时分频工作,也可以在正常工作时分频

I/O 口结构

I/O 口配置

STC12C5410AD 系列单片机其所有 I/O 口均可由软件配置成 4 种工作类型之一，如下表所示。4 种类型分别为：准双向口（标准 8051 输出模式)、推挽输出、仅为输入（高阻）或开漏输出功能。每个口由 2 个控制寄存器中的相应位控制每个引脚工作类型。STC12C5410AD 系列单片机上电复位后为准双向口（标准 8051 输出模式）模式。2V 以上时为高电平，0.8V 以下时为低电平。

I/O 口工作类型设定

P3 口设定 <P3.7,P3.6,P3.5,P3.4,P3.3,P3.2,P3.1,P3.0>

P3M0【7：0】	P3M1【7：0】	I/O 口模式
0	0	准双向口(传统 8051 I/O 口模式）， 灌电流可达 20mA，拉电流为 230μA， 由于制造误差，实际为 250uA～160uA
0	1	推挽输出（强上拉输出，可达 20mA，尽量少用）
1	0	仅为输入（高阻）
1	1	开漏(Open Drain)，内部上拉电阻断开，要外加

P2 口设定 <P2.7,P2.6,P2.5,P2.4,P2.3,P2.2,P2.1,P2.0>

P2M0【7：0】	P2M1【7：0】	I/O 口模式
0	0	准双向口(传统 8051 I/O 口模式）， 灌电流可达 20mA，拉电流为 230μA， 由于制造误差，实际为 250uA～160uA
0	1	推挽输出（强上拉输出，可达 20mA，尽量少用）
1	0	仅为输入（高阻）
1	1	开漏(Open Drain)，内部上拉电阻断开，要外加

P1 口设定 <P1.7,P1.6,P1.5,P1.4,P1.3,P1.2,P1.1,P1.0>

P1M0【7：0】	P1M1【7：0】	I/O 口模式（P1.x 如做A/D使用，需先将其设置成开漏或高阻输入）
0	0	准双向口（传统8051 I/O 口模式）， 灌电流可达20mA，拉电流为230μA， 由于制造误差，实际为250uA～160uA
0	1	推挽输出（强上拉输出，可达20mA，尽量少用）
1	0	仅为输入（高阻），如果该I/O口需作为A/D使用，可选此模式
1	1	开漏(Open Drain)，如果该I/O口需作为A/D使用，可选此模式

P0 口设定 <P0.7,P0.6,P0.5,P0.4,P0.3,P0.2,P0.1,P0.0>

P0M0【7：0】	P0M1【7：0】	I/O 口模式
0	0	准双向口(传统 8051 I/O 口模式）， 灌电流可达 20mA，拉电流为 230μA， 由于制造误差，实际为 250uA～160uA
0	1	推挽输出（强上拉输出，可达 20mA，尽量少用）
1	0	仅为输入（高阻）
1	1	开漏(Open Drain)，内部上拉电阻断开，要外加

举例:

```
MOV   P1M0,#11000000B
MOV   P1M1,#10100000B
;P1.7 为开漏,P1.6 为高阻输入,P1.5 为强推挽输出,P1.4/P1.3/P1.2/P1.1/P1.0 为弱上拉
```

注意:

虽然每个 I/O 口在弱上拉时都能承受 20mA 的灌电流(还是要加限流电阻,如 1K，560 Ω等),在强推挽输出时都能输出 20mA 的拉电流,但是整个芯片的工作电流推荐不要超过 55mA。

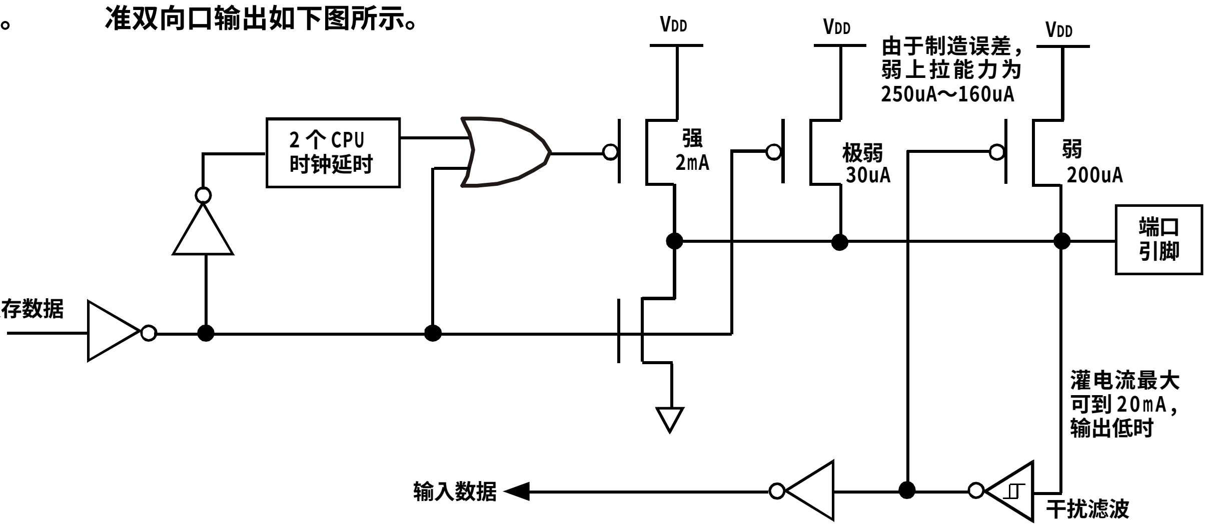
1.准双向口输出配置

准双向口输出类型可用作输出和输入功能而不需重新配置口线输出状态。这是因为当口线输出为1时驱动能力很弱，允许外部装置将其拉低。当引脚输出为低时，它的驱动能力很强，可吸收相当大的电流。准双向口有3个上拉晶体管适应不同的需要。

在3个上拉晶体管中，有1个上拉晶体管称为"弱上拉"，当口线寄存器为1且引脚本身也为1时打开。此上拉提供基本驱动电流使准双向口输出为1。如果一个引脚输出为1而由外部装置下拉到低时，弱上拉关闭而"极弱上拉"维持开状态，为了把这个引脚强拉为低，外部装置必须有足够的灌电流能力使引脚上的电压降到门槛电压以下。

第2个上拉晶体管，称为"极弱上拉"，当口线锁存为1时打开。当引脚悬空时，这个极弱的上拉源产生很弱的上拉电流将引脚上拉为高电平。

第3个上拉晶体管称为"强上拉"。当口线锁存器由0到1跳变时，这个上拉用来加快准双向口由逻辑0到逻辑1转换。当发生这种情况时，强上拉打开约2个机器周期以使引脚能够迅速地上拉到高电平。准双向口输出如下图所示。



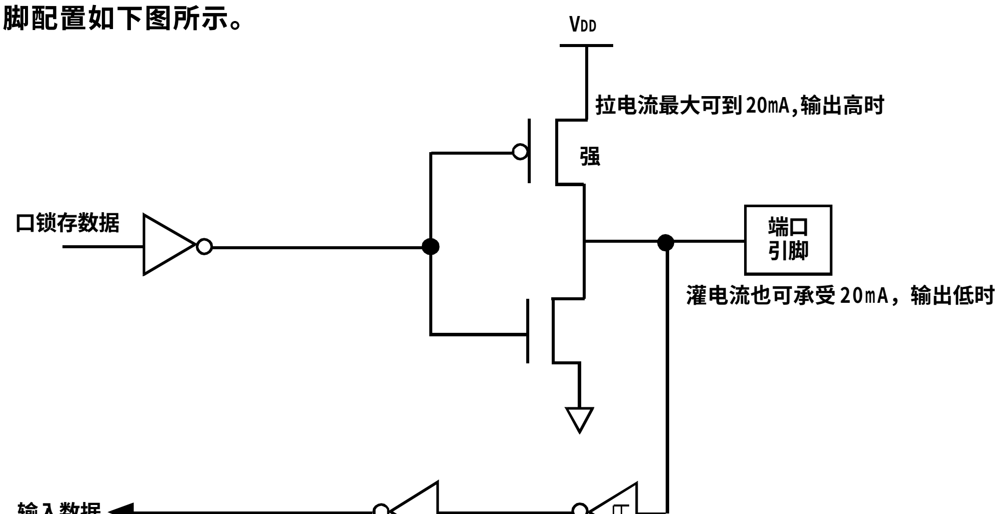
STC12LE5410系列单片机为3V器件，如果用户在引脚加上5V电压，将会有电流从引脚流向VDD，这样导致额外的功率消耗。因此，建议不要在准双向口模式中向3V单片机引脚施加5V电压,如使用的话，要加限流电阻，或用二极管做输入隔离，或用三极管做输出隔离。

准双向口带有一个施密特触发输入以及一个干扰抑制电路。

2.推挽输出配置

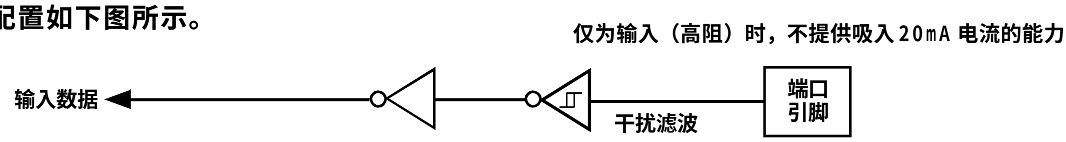
推挽输出配置的下拉结构与开漏输出以及准双向口的下拉结构相同，但当锁存器为1时提供持续的强上拉。推挽模式一般用于需要更大驱动电流的情况。

推挽引脚配置如下图所示。



3．仅为输入（高阻）配置

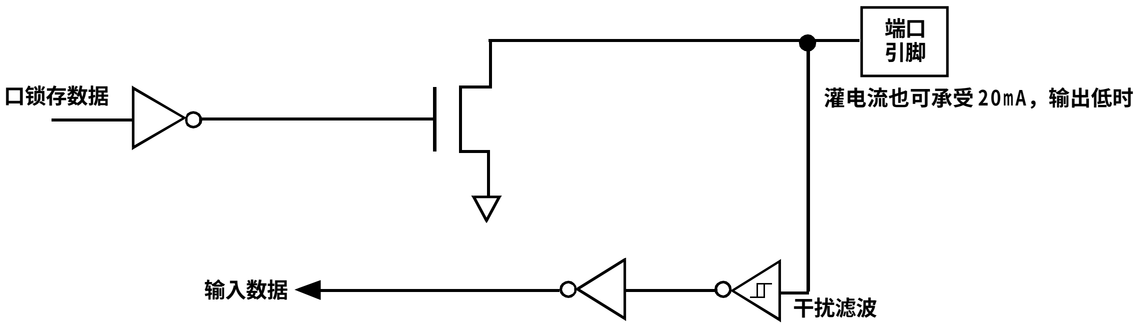
输入口配置如下图所示。



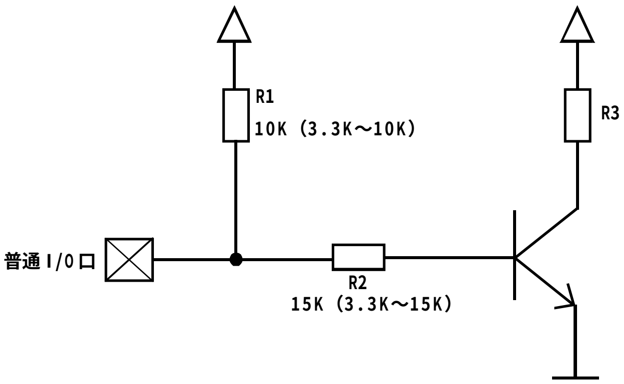
输入口带有一个施密特触发输入以及一个干扰抑制电路。

4.开漏输出配置

当口线锁存器为0时，开漏输出关闭所有上拉晶体管。当作为一个逻辑输出时，这种配置方式必须有外部上拉，一般通过电阻外接到VDD。这种方式的下拉与准双向口相同。输出口线配置如下图所示。开漏端口带有一个施密特触发输入以及一个干扰抑制电路。



一种典型三极管控制电路



如果用弱上拉控制，建议加上拉电阻R1（3.3K～10K），如果不加上拉电阻R1（3.3K～10K），建议R2的值在15K以上，或用强推挽输出

A/D 及 A/D 转换控制寄存器 ADC_CONTR/ADC_DATA

STC12C5410AD 系列带 A/D 转换的单片机在 P1 口，有 8 路 10 位高速 A/D 转换器， 速度可达 100KHz。P1.7 - P1.0 共 8 路电压输入型 A/D，可做温度检测、电池电压检测、按键扫描、频谱检测等。上电复位后 P1 口为弱上拉型 I/O 口，用户可以通过软件设置将 8 路中的任何一路设置为 A/D 转换，不需作为 A/D 使用的口可继续作为 I/O 口使用。需作为 A/D 使用的口需先将其设置为高阻输入或开漏模式。在 P1M0、P1M1 中对相应的位进行设置。

P1M0【7：0】地址：91h	P1M1【7：0】地址：92h	I/O 口模式（P1.x 如做A/D使用，需先将其设置成开漏或高阻输入）
0	0	准双向口（传统8051 I/O 口模式），灌电流可达20mA，拉电流为230μA
0	1	推挽输出（强上拉输出，可达20mA，尽量少用）
1	0	仅为输入（高阻），如果该I/O口需作为A/D使用，可选此模式
1	1	开漏(Open Drain)，如果该I/O口需作为A/D使用，可选此模式

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
ADC_CONTR	C5h	A/D 转换控制寄存器	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	0xx0,0000
ADC_DATA	C6h	A/D 转换结果寄存器,高8位	-	-	-	-	-	-	-	-	xxxx,xxxx
ADC_LOW2	BEh	A/D 转换结果寄存器，低2位	-	-	-	-	-	-	-	-	xxxx,xxxx

ADC_CONTR 特殊功能寄存器：A/D 转换控制特殊功能寄存器

A/D 转换控制寄存器	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	0xx0,0000

CHS2 / CHS1 / CHS0：模拟输入通道选择，CHS2 / CHS1 / CHS0

CHS2	CHS1	CHS0	Analog Input Channel Select 模拟输入通道选择
0	0	0	选择 P1.0 作为 A/D 输入来用
0	0	1	选择 P1.1 作为 A/D 输入来用
0	1	0	选择 P1.2 作为 A/D 输入来用
0	1	1	选择 P1.3 作为 A/D 输入来用
1	0	0	选择 P1.4 作为 A/D 输入来用
1	0	1	选择 P1.5 作为 A/D 输入来用
1	1	0	选择 P1.6 作为 A/D 输入来用
1	1	1	选择 P1.7 作为 A/D 输入来用

ADC_START: 模数转换器(ADC)转换启动控制位，设置为“1”时，开始转换，转换结束后为 0。

ADC_FLAG: 模数转换器转换结束标志位，当 A/D 转换完成后，ADC_FLAG = 1，要由软件清 0。
不管是 A/D 转换完成后由该位申请产生中断，还是由软件查询该标志位 A/D 转换是否结束，当 A/D 转换完成后，ADC_FLAG = 1，一定要软件清 0。

SPEED1，SPEED0：模数转换器转换速度控制位

SPEED1	SPEED0	A/D 转换所需时间
1	1	210 个时钟周期转换一次，CPU 工作频率 20MHz 时，A/D 转换速度约 100KHz
1	0	420 个时钟周期转换一次
0	1	630 个时钟周期转换一次
0	0	840 个时钟周期转换一次

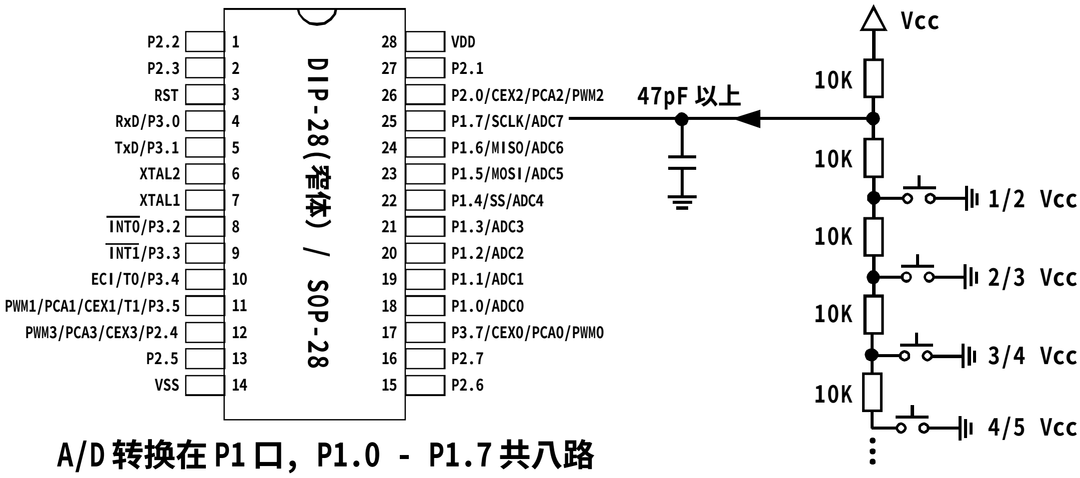
ADC_POWER: ADC 电源控制位。
0：关闭 ADC 电源；1：打开 A/D 转换器电源
启动 AD 转换前一定要确认 AD 电源已打开，AD 转换结束后关闭 AD 电源可降低功耗，也可不关闭。
初次打开内部 A/D 转换模拟电源，需适当延时，等内部模拟电源稳定后，再启动 A/D 转换
建议启动 A/D 转换后，在 A/D 转换结束之前，不改变任何 I/O 口的状态，有利于高精度 A/D 转换

ADC_DATA / ADC_LOW2 特殊功能寄存器：A/D 转换结果特殊功能寄存器

ADC_DATA	C6h	A/D 转换结果寄存器,全部8位有效，为10位A/D转换结果的高8位	-	-	-	-	-	-	-	-	xxxx,xxxx
ADC_LOW2	BEh	A/D 转换结果寄存器，只有低2位有效，为10位A/D转换结果的低2位	x	x	x	x	x	x	-	-	xxxx,xxxx

模拟/数字转换结果计算公式如下：结果 (ADC_DATA[7:0],ADC_LOW2[1:0]) = 1024 x Vin / Vcc
Vin 为模拟输入通道输入电压，Vcc 为单片机实际工作电压，用单片机工作电压作为模拟参考电压。
取 ADC_DATA 的 8 位为 ADC 转换的高 8 位，取 ADC_LOW2 的低 2 位为 ADC 转换的低 2 位，则为 10 位精度。
如果舍弃 ADC_LOW2 的低 2 位，只用 ADC_DATA 寄存器的 8 位，则 A/D 转换结果为 8 位精度。
结果 ADC_DATA[7:0] = 256 x Vin / Vcc

A/D 转换典型应用线路，按键扫描



A/D 转换在 P1 口，P1.0 - P1.7 共八路

A/D 转换功能汇编程序示例

```
;/* ---  STC International Limited ---------------- */
;/* ---  宏晶科技 姚永平  设计 2006/1/6   V1.0 ----- */
;/* ---  演示 STC12C5410AD 系列 MCU 的 A/D 转换功能- */
;/* ---  演示 STC12C2052AD 系列 MCU 的 A/D 转换功能- */
;/* ---  Mobile: 13922805190 ---------------------- */
;/* ---  Fax: 0755-82944243 ----------------------- */
;/* ---  Tel: 0755-82948409 ----------------------- */
;/* ---  Web: www.mcu-memory.com ------------------ */

;ADC DEMO_5410_ASM.ASM 汇编程序演示 STC12C5410AD 系列 MCU 的 A/D 转换功能。
;本演示程序在STC-ISP Ver 3.0A.PCB的下载编程工具上测试通过,相关的A/D转换结果在P1口上显示
;转换结果也以 16 进制形式输出到串行口，可以用串行口调试程序观察输出结果。
;时钟 18.432MHz, 波特率 = 9600。
;转换结果也在 P1 口利用 LED 显示出来，方便观察。

LED_MCU_START    EQU   P3.7
ADC_CONTR   EQU   0C5H     ;A/D转换寄存器
ADC_DATA    EQU   0C6H     ;A/D转换结果寄存器，为10位 A/D转换结果的高8位
;ADC_LOW2    EQU  0BEH     ;A/D转换结果寄存器，低2位有效，为10位 A/D转换结果的低2位
                            ;如果不用ADC_LOW2的低2位,只用ADC_DATA的8位,则为8位A/D转换
P1M0        EQU   91H      ;P1口模式选择寄存器0
P1M1        EQU   92H      ;P1口模式选择寄存器1

ADC_Power_On_Speed_Channel_0 EQU 11100000B     ;P1.0作为A/D输入
ADC_Power_On_Speed_Channel_1 EQU 11100001B     ;P1.1作为A/D输入
ADC_Power_On_Speed_Channel_2 EQU 11100010B     ;P1.2作为A/D输入
ADC_Power_On_Speed_Channel_3 EQU 11100011B     ;P1.3作为A/D输入
ADC_Power_On_Speed_Channel_4 EQU 11100100B     ;P1.4作为A/D输入
ADC_Power_On_Speed_Channel_5 EQU 11100101B     ;P1.5作为A/D输入
ADC_Power_On_Speed_Channel_6 EQU 11100110B     ;P1.6作为A/D输入
ADC_Power_On_Speed_Channel_7 EQU 11100111B     ;P1.7作为A/D输入

;-------------------------------------------------
;定义变量
ADC_Channel_0_Result      EQU 30H      ;0通道A/D转换结果
ADC_Channel_1_Result      EQU 31H      ;1通道A/D转换结果
ADC_Channel_2_Result      EQU 32H      ;2通道A/D转换结果
ADC_Channel_3_Result      EQU 33H      ;3通道A/D转换结果
ADC_Channel_4_Result      EQU 34H      ;4通道A/D转换结果
ADC_Channel_5_Result      EQU 35H      ;5通道A/D转换结果
ADC_Channel_6_Result      EQU 36H      ;6通道A/D转换结果
ADC_Channel_7_Result      EQU 37H      ;7通道A/D转换结果
```

```
;------------------------------------------------
    ORG    0000H
    LJMP   MAIN

    ORG    0050H
MAIN:
    CLR    LED_MCU_START              ;MCU 工作指示灯 LED_MCU_START   EQU P3.7
    MOV    SP, #7FH                   ;设置堆栈

    ACALL Initiate_RS232              ;初始化串口

    ACALL ADC_Power_On                ;开 ADC 电源，第一次使用时要打开内部模拟电源
                                      ;开 ADC 电源，可适当加延时，1mS 以内就足够了
    ACALL Set_P12_Open_Drain          ;设置 P1.2 为开漏
    ACALL Set_ADC_Channel_2           ;设置 P1.2 作为 A/D 转换通道

    ACALL Get_AD_Result               ;测量电压并且取 A/D 转换结果
    ACALL Send_AD_Result              ;发送转换结果到 PC 机

    ACALL Set_P12_Normal_IO           ;设置 P1.2 为普通 IO
    MOV    A, ADC_Channel_2_Result    ;用 P1 口显示 A/D 转换结果
    CPL    A
    MOV    P1, A

Wait_Loop:
    SJMP   Wait_Loop                  ;停机

;-----------------------------------------------------------
;-----------------------------------------------------------
;-----------------------------------------------------------
;-----------------------------------------------------------
Initiate_RS232:                       ;串口初始化
    CLR    ES                         ;禁止串口中断
    MOV    TMOD, #20H                 ;设置 T1 为波特率发生器
    MOV    SCON, #50H                 ;0101,0000 8 位数据位，无奇偶校验
    MOV    TH1, #0FBH                 ;18.432MHz 晶振，波特率 = 9600
    MOV    TL1, #0FBH

    SETB   TR1                        ;启动 T1
    RET
;------------------------------------------------------------
Send_Byte:
    CLR    TI
    MOV    SBUF, A
Send_Byte_Wait_Finish:
    JNB    TI, Send_Byte_Wait_Finish
    CLR    TI
    RET
```

```
;-------------------------------------------------------------------
ADC_Power_On:
    PUSH  ACC
    ORL   ADC_CONTR, #80H           ;开A/D转换电源
    MOV   A, #20H
    ACALL Delay                     ;开A/D转换电源后要加延时，1mS以内就足够了
    POP   ACC
    RET

;-------------------------------------------------------------------
;设置P1.2,设置A/D通道所在的I/O为开漏模式
Set_P12_Open_Drain:
    PUSH  ACC
    MOV   A, #00000100B
    ORL   P1M0, A
    ORL   P1M1, A
    POP   ACC
    RET

;-------------------------------------------------------------------
;设置 P1.2 为普通IO
Set_P12_Normal_IO:
    PUSH  ACC
    MOV   A, #11111011B
    ANL   P1M0, A
    ANL   P1M1, A
    POP   ACC
    RET

;-------------------------------------------------------------------
Set_ADC_Channel_2:
    MOV   ADC_CONTR, #ADC_Power_On_Speed_Channel_2
                  ;选择P1.2作为A/D转换通道
    MOV   A, #05H ;更换 A/D 转换通道后要适当延时，使输入电压稳定
                  ;以后如果不更换 A/D 转换通道的话，不需要加延时
    ACALL Delay   ;切换 A/D 转换通道,加延时20uS～200uS就可以了,与输入电压源的内阻有关
    RET

;-------------------------------------------------------------------

Send_AD_Result:
    PUSH  ACC
    MOV   A, ADC_Channel_2_Result   ;取 AD 转换结果
    ACALL Send_Byte                 ;发送转换结果到 PC 机
    POP   ACC
    RET
```

```
;-----------------------------------------------------------
Get_AD_Result:
    PUSH  ACC                          ;入栈保护
    MOV   ADC_DATA, #0
    ORL   ADC_CONTR, #00001000B        ;启动 AD 转换

Wait_AD_Finishe:
    MOV   A, #00010000B                ;判断 AD 转换是否完成
    ANL   A, ADC_CONTR
    JZ    Wait_AD_Finishe              ;AD 转换尚未完成，继续等待

    ANL   ADC_CONTR, #11100111B        ;清 0 ADC_FLAG, ADC_START 位，停止 A/D 转换

    MOV   A, ADC_DATA
    MOV   ADC_Channel_2_Result, A      ;保存 AD 转换结果
    POP   ACC
    RET
;-----------------------------------------------------------------
Delay:
    PUSH  02                           ;将寄存器组 0 的 R2 入栈
    PUSH  03                           ;将寄存器组 0 的 R3 入栈
    PUSH  04                           ;将寄存器组 0 的 R4 入栈
    MOV   R4, A

Delay_Loop0:
    MOV   R3, #200              ;2 CLOCK  ---------------------+
Delay_Loop1:                    ;                              |
    MOV   R2, #249              ;2 CLOCK  ------+              |
Delay_Loop:                     ;               | 1002 CLOCK   |200406 CLOCK
    DJNZ  R2, Delay_Loop        ;4 CLOCK        |              |
    DJNZ  R3, Delay_Loop1       ;4 CLOCK  ------+              |
    DJNZ  R4, Delay_Loop0       ;4 CLOCK  ---------------------+

    POP   04
    POP   03
    POP   02
    RET
;-----------------------------------------------------------------
    END
```

看门狗应用

适用型号： STC12C5410AD系列

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
WDT_CONTR	E1h	Watch-Dog-Timer Control register	WDT_FLAG	-	EN_WDT	CLR_WDT	IDLE_WDT	PS2	PS1	PS0	xx00,0000

Symbol 符号 Function 功能

WDT_FLAG When WDT overflows, this bit is set. It can be cleared by software.
看门狗溢出标志位,当溢出时，该位由硬件置1，可用软件将其清0。

EN_WDT Enable WDT bit. When set, WDT is started
看门狗允许位，当设置为“1”时，看门狗启动。

CLR_WDT WDT clear bit. When set, WDT will recount. Hardware will automatically clear this bit.
看门狗清“0”位,当设为“1”时，看门狗将重新计数。硬件将自动清“0”此位。

IDLE_WDT When set, WDT is enabled in IDLE mode. When clear, WDT is disabled in IDLE mode
看门狗“IDLE”模式位,当设置为“1”时,看门狗定时器在“空闲模式”计数
当清“0”该位时，看门狗定时器在“空闲模式”时不计数

PS2, PS1, PS0 Pre-scale value of Watchdog timer is shown as the bellowed table:
看门狗定时器预分频值，如下表所示

PS2	PS1	PS0	Pre-scale 预分频	WDT Period @20MHz
0	0	0	2	39.3 mS
0	0	1	4	78.6 mS
0	1	0	8	157.3 mS
0	1	1	16	314.6 mS
1	0	0	32	629.1 mS
1	0	1	64	1.25S
1	1	0	128	2.5S
1	1	1	256	5S

The WDT period is determined by the following equation 看门狗溢出时间计算

看门狗溢出时间 = (12 x Pre-scale x 32768) / Oscillator frequency

设时钟为12MHz：

看门狗溢出时间 = (12 x Pre-scale x 32768) / 12000000 = Pre-scale x 393216 / 12000000

PS2	PS1	PS0	Pre-scale 预分频	WDT Period @12MHz
0	0	0	2	65.5 mS
0	0	1	4	131.0 mS
0	1	0	8	262.1 mS
0	1	1	16	524.2 mS
1	0	0	32	1.0485S
1	0	1	64	2.0971S
1	1	0	128	4.1943S
1	1	1	256	8.3886S

设时钟为 11.0592MHz：

看门狗溢出时间 = (12 x Pre-scale x 32768) / 11059200 = Pre-scale x 393216 / 11059200

PS2	PS1	PS0	Pre-scale 预分频	WDT Period @11.0592MHz
0	0	0	2	71.1 mS
0	0	1	4	142.2 mS
0	1	0	8	284.4 mS
0	1	1	16	568.8 mS
1	0	0	32	1.1377S
1	0	1	64	2.2755S
1	1	0	128	4.5511S
1	1	1	256	9.1022S

汇编语言程序示例

```
WDT_CONTR   DATA    0E1H ;      或者    WDT_CONTR  EQU    0E1H
;复位入口
    ORG     0000H
    LJMP    Initial
    …
    ORG     0060H
Initial:
    MOV     WDT_CONTR,  #00111100B; Load initial value 看门狗定时器控制寄存器初始化
                ; EN_WDT = 1, CLR_WDT = 1, IDLE_WDT = 1, PS2 = 1, PS1 = 0, PS0 = 0
    …
Main_Loop:
    LCALL   Display_Loop
    LCALL   Keyboard_Loop
    …
    MOV     WDT_CONTR,  #00111100B ;  喂狗， 不要用 ORL    WDT_CONTR,  #00010000B
    …
    LJMP    Main_Loop
```

C 语言程序示例

```
#include<reg52.h>
sfr     WDT_CONTR   =   0xe1;
void main()
{   …
    WDT_CONTR   =   0x3c;
       /* 0011,1100 EN_WDT = 1,CLR_WDT = 1,IDLE_WDT = 1,PS2 = 1,PS1 = 0,PS0 = 0 */
    while(1){
        display();
        keyboard();
        …
        WDT_CONTR   =   0x3c;  /* 喂狗， 不要用  WDT_CONTR  =   WDT_CONTR | 0x10;*/
    }
}
```

;本程序验证 STC12C5410AD 系列及STC12C2052AD 系列单片机的看门狗及其溢出时间计算公式

```
;本演示程序在STC-ISP Ver 3.0A.PCB的下载编程工具上测试通过,相关的工作状态在P1口上显示
;看门狗及其溢出时间 = (12 * Pre_scale *32768)/Oscillator frequency
WDT_CONTR        EQU 0E1H ;看门狗地址
WDT_TIME_LED     EQU P1.5 ;用 P1.5 控制看门狗溢出时间指示灯,
                          ;看门狗溢出时间可由该指示灯亮的时间长度或熄灭的时间长度表示
WDT_FLAG_LED     EQU P1.7 ;用 P1.7 控制看门狗溢出复位指示灯, 如点亮表示为看门狗溢出复位
Last_WDT_Time_LED_Status  EQU 00H ;位变量, 存储看门狗溢出时间指示灯的上一次状态位
;WDT 复位时间(所用的Oscillator frequency = 18.432MHz):
;Pre_scale_Word  EQU 00111100B   ;清0,启动看门狗, 预分频数=32,   0.68S
Pre_scale_Word  EQU 00111101B    ;清0,启动看门狗, 预分频数=64,   1.36S
;Pre_scale_Word  EQU 00111110B   ;清0,启动看门狗, 预分频数=128,  2.72S
;Pre_scale_Word  EQU 00111111B   ;清0,启动看门狗, 预分频数=256,  5.44S
    ORG    0000H
    AJMP   MAIN
    ORG    0100H
MAIN:
    MOV    A, WDT_CONTR             ;检测是否为看门狗复位
    ANL    A, #10000000B
    JNZ    WDT_Reset                ;WDT_CONTR.7 = 1, 看门狗复位, 跳转到看门狗复位程序
;WDT_CONTR.7 = 0,上电复位, 冷启动, RAM 单元内容为随机值
    SETB   Last_WDT_Time_LED_Status        ;上电复位,
                                           ;初始化看门狗溢出时间指示灯的状态位 = 1
    CLR    WDT_TIME_LED                    ;上电复位, 点亮看门狗溢出时间指示灯
    MOV    WDT_CONTR, #Pre_scale_Word      ;启动看门狗
WAIT1:
    SJMP   WAIT1                    ;循环执行本语句(停机), 等待看门狗溢出复位

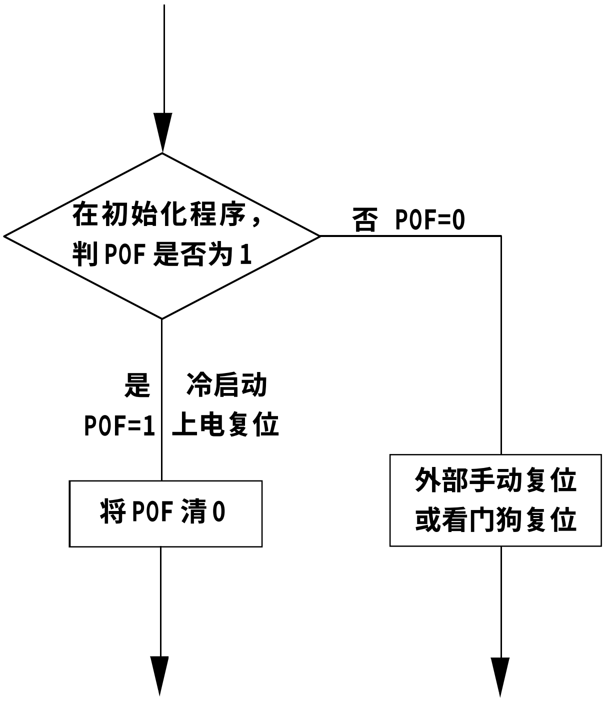
;WDT_CONTR.7 = 1,看门狗复位, 热启动, RAM 单元内容不变, 为复位前的值
WDT_Reset:                          ;看门狗复位, 热启动
    CLR    WDT_FLAG_LED             ;是看门狗复位,点亮看门狗溢出复位指示灯

    JB    Last_WDT_Time_LED_Status, Power_Off_WDT_TIME_LED;为1熄灭相应的灯,为0亮相应灯
    ;根据看门狗溢出时间指示灯的上一次状态位设置 WDT_TIME_LED 灯,
    ;若上次亮本次就熄灭, 若上次熄灭本次就亮
    CLR    WDT_TIME_LED             ;上次熄灭本次点亮看门狗溢出时间指示灯
    CPL    Last_WDT_Time_LED_Status ;将看门狗溢出时间指示灯的上一次状态位取反
WAIT2:
    SJMP   WAIT2                    ;循环执行本语句(停机), 等待看门狗溢出复位
Power_Off_WDT_TIME_LED:
    SETB   WDT_TIME_LED             ;上次亮本次就熄灭看门狗溢出时间指示灯
    CPL    Last_WDT_Time_LED_Status ;将看门狗溢出时间指示灯的上一次状态位取反
WAIT3:
    SJMP   WAIT3                    ;循环执行本语句(停机), 等待看门狗溢出复位
    END
```

PCON 寄存器的高级应用，上电复位标志，进入掉电模式

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
PCON	87h	Power Control	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000

POF：上电复位标志位，单片机停电后，上电复位标志位为 1，可由软件清 0。
实际应用：要判断是上电复位（冷启动），还是外部复位脚输入复位信号产生的复位，还是内部看门狗复位，可通过如下方法来判断：



PD：将其置 1 时，进入 Power Down 模式，可由外部中断低电平触发或下降沿触发中断模式唤醒。
进入掉电模式时，外部时钟停振，CPU、定时器、串行口全部停止工作，只有外部中断继续工作。
IDL：将其置 1 时，进入 IDLE 模式（空闲），除 CPU 不工作外，其余仍继续工作，可由任何一个中断唤醒。
现 C 版本开始大量供货，C 版本 IDLE 模式可正常使用（原 A 版本和 B 版本建议不要用 IDLE 模式）。

STC12C5410AD 系列 1T 单片机通过外部中断从掉电模式唤醒

```
;*************************************************************************
;Wake Up Idle and Wake Up Power Down
;*************************************************************************
    ORG     0000H
    AJMP  MAIN

    ORG     0003H
 int0_interrupt:
    CLR     P1.7              ;点亮 P1.7 LED 表示已响应 int0 中断
    ACALL  delay              ;延时是为了便于观察，实际应用不需延时
    CLR     EA                ;关闭中断，简化实验. 实际应用不需关闭中断
    RETI

    ORG     0013H
 int1_interrupt:
    CLR     P1.6              ;点亮 P1.6 LED 表示已响应 int1 中断
    ACALL  delay              ;延时是为了便于观察，实际应用不需延时
    CLR     EA                ;关闭中断，简化实验. 实际应用不需关闭中断
    RETI

    ORG     0100H
 delay:
    CLR     A
    MOV     R0, A
    MOV     R1, A
    MOV     R2, #02
 delay_loop:
    DJNZ   R0, delay_loop
    DJNZ   R1, delay_loop
    DJNZ   R2, delay_loop
    RET

 main:
    MOV     R3, #0            ;P1 LED 递增方式变化，表示程序开始运行
 main_loop:
    MOV     A, R3
    CPL     A
    MOV     P1, A
    ACALL  delay
```

```
    INC     R3
    MOV     A, R3
    SUBB    A, #18H
    JC      main_loop

    MOV     P1, #0FFH       ;熄灭全部灯表示进入 Power Down 状态

    CLR     IT0             ;设置低电平激活外部中断
;   SETB    IT0

    SETB    EX0             ;允许外部中断 0

    CLR     IT1             ;设置低电平激活外部中断
;   SETB    IT1
    SETB    EX1             ;允许外部中断 1
;   SETB    ET0    ;如果是 STC12C2052AD 系列 A 版本,
                   ;要由外部中断 1 唤醒,“ET0=1”是必须的,硬件就这样做的,C 版本就不需要
                   ;外部中断 0 就无此必要,建议 Powerdown 用外部中断 0 唤醒

    SETB    EA              ;开中断，若不开中断就不能唤醒 Power Down

;下条语句将使 MCU 进入 idle 状态或 Power Down 状态
;低电平激活外部中断可以将 MCU 从 Power Down 状态中唤醒
;其方法为:将外部中断脚拉低

    MOV     A, PCON         ;令 PD=1, 进入 Power Down 状态, PD = PCON.2
    ORL     A, #02H
    MOV     PCON, A

    MOV     PCON, #01H      ;删除本语句前的";", 同时将前 3 条语句前加上注释符号";",
                            ;令 IDL=1, 可进入 idle 状态, IDL = PCON.1

    MOV     P1, #0DFH       ;请注意:
                            ; 1.外部中断使 MCU 退出 Power Down 状态,执行本条指令后
                            ;响应中断，表现为 P1.5 与 P1.7 的 LED 同时亮(INT0 唤醒)
                            ; 2.外部中断使 MCU 退出 idle 状态,先响应中断然后再执行本
                            ;条指令，表现为 P1.7 的 LED 先亮(INT0 唤醒)P1.5 的 LED 后亮
WAIT1:
    SJMP    WAIT1           ;跳转到本语句，停机
    END
;A 版本和 B 版本建议不要用 IDLE 模式,现 C 版本可以正常使用
```

STC12C5410AD 系列单片机进入掉电模式前的 I/O 设置特别注意事项 --- 仅针对 A 版本，现供货的是最新版本 C 版本

※ STC12C5410AD 系列单片机最新的 C 版本已开始大量供货，已解决此问题，B 版本也无问题，只有 A 版本需要注意

针对 A 版本如下应用注意事项：
STC12Cxx 要 Power Down < 0.1uA，进入 Power Down 之前
1.
如果您系统外部所接的口是输入，我们 STC12Cxx 是输出，
请将我们的该 I/O 设为 Push-Pull 输出
外部是低，你就要为低；外部是高，你就要为高，否则就 > 0.1uA
如果是后续的版本，不一定要设置成 Push-Pull 输出，用弱上拉就可以了。

2.
如果您系统外部所接的口是输出，我们 STC12Cxx 是输入，
请将我们的该 I/O 设为 Only-Input，高阻仅为输入
如果是后续的版本，不一定要设置成 Only-Input 输出，用弱上拉就可以了。

3.
如果您系统外部所接的口是高阻，我们 STC12Cxx ，
请将我们的该 I/O 设为 Push-Pull 输出
请不要将我们的该 I/O 设为 Only-Input/高阻
如果是后续的版本，不一定要设置成 Push-Pull 输出，用弱上拉就可以了。

现在 C 版本已开始供货，定货时指定 C 版本就可以了

STC12C5410AD 系列 1T 8051 单片机 IAP 应用
STC12C5410AD 系列 1T 8051 单片机内部 EEPROM 的应用

-- 利用 IAP 技术可实现 EEPROM，内部 Flash 擦写次数为 100,000 次以上

STC12C5410AD 系列 1T 8051 单片机 ISP/IAP 特殊功能寄存器 ISP/IAP SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
ISP_DATA	E2h	ISP/IAP Flash Data Register									1111,1111
ISP_ADDRH	E3h	ISP/IAP Flash Address High									0000,0000
ISP_ADDRL	E4h	ISP/IAP Flash Address Low									0000,0000
ISP_CMD	E5h	ISP/IAP Flash Command Register	-	-	-	-	-	-	MS1	MS0	xxxx,xx00
ISP_TRIG	E6h	ISP/IAP Flash Command Trigger									xxxx,xxxx
ISP_CONTR	E7h	ISP/IAP Control Register	ISPEN	SWBS	SWRST	CMD_FAIL	1	WT2	WT1	WT0	0000,1000

ISP_DATA: ISP/IAP 操作时的数据寄存器。
ISP/IAP 从 Flash 读出的数据放在此处，向 Flash 写的数据也需放在此处

ISP_ADDRH: ISP/IAP 操作时的地址寄存器高八位。

ISP_ADDRL: ISP/IAP 操作时的地址寄存器低八位。

ISP_CMD: ISP/IAP 操作时的命令模式寄存器，须命令触发寄存器触发方可生效。

B7	B6	B5	B4	B3	B2	B1	B0	命令 / 操作 模式选择
保留						命令		
-	-	-	-	-	-	0	0	Standby 待机模式，无ISP操作
-	-	-	-	-	-	0	1	从用户的应用程序区对 "Data Flash/EEPROM区" 进行字节读
-	-	-	-	-	-	1	0	从用户的应用程序区对 "Data Flash/EEPROM区" 进行字节编程
-	-	-	-	-	-	1	1	从用户的应用程序区对 "Data Flash/EEPROM区" 进行扇区擦除

程序在用户应用程序区时，仅可以对数据 Flash 区(EEPROM)进行字节读 / 字节编程 / 扇区擦除，STC12C5412AD/STC12LE5412AD 除外。STC12C5410AD 系列单片机出厂时就已完全加密。

ISP_TRIG: ISP/IAP 操作时的命令触发寄存器。
在 ISPEN(ISP_CONTR.7) = 1时,对 ISP_TRIG 先写入 46h,再写入 B9h,
ISP/IAP 命令才会生效。

ISP_CONTR: ISP/IAP 控制寄存器。

B7	B6	B5	B4	B3	B2	B1	B0	Reset Value
ISPEN	SWBS	SWRST	CMD_FAIL	1	WT2	WT1	WT0	0000,1000

ISPEN: ISP/IAP 功能允许位。0：禁止 ISP/IAP 编程改变 Flash,1:允许编程改变 Flash

SWBS:软件选择从用户主程序区启动（0），还是从 ISP 程序区启动（1）。

SWRST: 0: 不操作； 1: 产生软件系统复位，硬件自动清零。

CMD_FAIL: 如果送了 ISP/IAP 命令，并对 ISP_TRIG 送 46h/B9h 触发失败，则为 1，需由软件清零。

设置等待时间			CPU 等待时间(CPU 的工作时钟)			
WT2	WT1	WT0	Read	Program	Sector Erase	Recommended System Clock
1	1	1	2	55	21012	1MHz
1	1	0	2	110	42024	2MHz
1	0	1	2	165	63036	3MHz
1	0	0	2	330	126072	6MHz
0	1	1	2	660	252144	12MHz
0	1	0	2	1100	420240	20MHz
0	0	1	2	1320	504288	24MHz
0	0	0	2	1760	672384	30MHz

STC12C5410AD 系列单片机内部可用 Data Flash(EEPROM)的地址(与程序空间是分开的)：
如果对应用程序区进行 IAP 写数据，则该语句会被单片机忽略，继续执行下一句。
程序在用户应用程序区(AP 区)时,仅可以对 Data Flash(EEPROM)进行 IAP/ISP 操作。
但 STC12C5412AD/STC12LE5412AD 在应用程序区可以修改应用程序区(灵活)。

STC12C5410AD,STC12C5410,STC12LE5410AD,STC12LE5410
STC12C5408AD,STC12C5408,STC12LE5408AD,STC12LE5408
STC12C5406AD,STC12C5406,STC12LE5406AD,STC12LE5406
STC12C5404AD,STC12C5404,STC12LE5404AD,STC12LE5404
STC12C5402AD,STC12C5402,STC12LE5402AD,STC12LE5402
STC12C5401AD,STC12C5401,STC12LE5401AD,STC12LE5401
系列单片机内部可用 Data Flash(EEPROM)的地址：

第一扇区		第二扇区		每个扇区 512字节 共4个扇区 建议同一次修改的数据放在同一个扇区，不必用满，当然可全用，用满则为2K字节EEPROM。由于擦除是按扇区擦除，所以每个扇区用的越少越方便，256个字节以内较合理。
起始地址	结束地址	起始地址	结束地址	
2800h	29FFh	2A00h	2BFFh	
第三扇区		第四扇区		
起始地址	结束地址	起始地址	结束地址	
2C00h	2DFFh	2E00h	2FFFh	

STC12C5412,STC12C5412AD,STC12LE5412,STC12LE5412AD 单片机可对自身内部应用程序区进行 IAP/ISP 操作，故所有部分均可当 Data Flash(EEPROM)使用，其地址如下：

第一扇区		第二扇区		第三扇区		第四扇区		每个扇区 512字节 建议同一次修改的数据放在同一个扇区，不必用满，当然可全用
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
0000h	01FFh	0200h	03FFh	0400h	05FFh	0600h	07FFh	
第五扇区		第六扇区		第七扇区		第八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
0800h	09FFh	0A00h	0BFFh	0C00h	0DFFh	0E00h	0FFFh	
第九扇区		第十扇区		第十一扇区		第十二扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
1000h	11FFh	1200h	13FFh	1400h	15FFh	1600h	17FFh	
第十三扇区		第十四扇区		第十五扇区		第十六扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
1800h	19FFh	1A00h	1BFFh	1C00h	1DFFh	1E00h	1FFFh	
第十七扇区		第十八扇区		第十九扇区		第二十扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
2000h	21FFh	2200h	23FFh	2400h	25FFh	2600h	27FFh	
第二十一扇区		第二十二扇区		第二十三扇区		第二十四扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
2800h	29FFh	2A00h	2BFFh	2C00h	2DFFh	2E00h	2FFFh	

STC12C5410AD 系列 IAP 应用汇编简介
STC12C5410AD 系列 内部 EEPROM 的应用

```
;用 DATA 还是 EQU 声明新增特殊功能寄存器地址要看你用的汇编器 / 编译器
    ISP_DATA          DATA      0E2h;或   ISP_DATA          EQU       0E2h
    ISP_ADDRH         DATA      0E3h;或   ISP_ADDRH         EQU       0E3h
    ISP_ADDRL         DATA      0E4h;或   ISP_ADDRL         EQU       0E4h
    ISP_CMD           DATA      0E5h;或   ISP_CMD           EQU       0E5h
    ISP_TRIG          DATA      0E6h;或   ISP_TRIG          EQU       0E6h
    ISP_CONTR         DATA      0E7h;或   ISP_CONTR         EQU       0E7h
;定义 ISP/IAP 命令及等待时间
    ISP_IAP_BYTE_READ      EQU   1     ;字节读
    ISP_IAP_BYTE_PROGRAM   EQU   2     ;字节编程,前提是该字节是空，0FFh
    ISP_IAP_SECTOR_ERASE   EQU   3     ;扇区擦除,要某字节为空，要擦一扇区
    WAIT_TIME         EQU   0          ;设置等待时间,30MHz 以下 0,24M 以下 1,
              ;20MHz 以下 2,12M 以下 3,6M 以下 4,3M 以下 5,2M 以下 6,1M 以下 7,
;字节读
    MOV   ISP_ADDRH, #BYTE_ADDR_HIGH        ;送地址高字节  ┐
    MOV   ISP_ADDRL, #BYTE_ADDR_LOW         ;送地址低字节  ┘ 地址需要改变时才需重新送地址
    MOV   ISP_CONTR, #WAIT_TIME ;设置等待时间          ┐
    ORL   ISP_CONTR, #10000000B ;允许 ISP/IAP 操作     ┘ 此两句可以合成一句,并且只送一次就够了
    MOV   ISP_CMD,        #ISP_IAP_BYTE_READ;送字节读命令,命令不需改变时,不需重新送命令
    MOV   ISP_TRIG,       #46h        ;先送 46h,再送 B9h 到 ISP/IAP 触发寄存器,每次都需如此
    MOV   ISP_TRIG,       #0B9h;送完 B9h 后，ISP/IAP 命令立即被触发起动
;CPU 等待 IAP 动作完成后，才会继续执行程序。
    NOP                                ;数据读出到 ISP_DATA 寄存器后，CPU 继续执行程序
    MOV   A,          ISP_DATA         ;将读出的数据送往 Acc
;以下语句可不用,只是出于安全考虑而已
    MOV   ISP_CONTR, #00000000B        ;禁止 ISP/IAP 操作
    MOV   ISP_CMD,   #00000000B        ;去除 ISP/IAP 命令
    MOV   ISP_TRIG,  #00000000B        ;防止 ISP/IAP 命令误触发
    ;MOV  ISP_ADDRH, #0  ;送地址高字节单元为 00,指向非 EEPROM 区
    ;MOV  ISP_ADDRL, #0  ;送地址低字节单元为 00,防止误操作
```

```
;字节编程，该字节为FFh/空时，可对其编程，否则不行,要先执行扇区擦除
    MOV   ISP_DATA,  #ONE_DATA     ;送字节编程数据到ISP_DATA,只有数据改变时才需重新送
    MOV   ISP_ADDRH, #BYTE_ADDR_HIGH ;送地址高字节  ┐
    MOV   ISP_ADDRL, #BYTE_ADDR_LOW  ;送地址低字节  ┘ 地址需要改变时才需重新送地址
    MOV   ISP_CONTR, #WAIT_TIME ;设置等待时间       ┐
    ORL   ISP_CONTR, #10000000B ;允许ISP/IAP操作    ┘ 此两句可合成一句,并且只送一次就够了
    MOV   ISP_CMD,   #ISP_IAP_BYTE_PROGRAM        ;送字节编程命令
    MOV   ISP_TRIG,  #46h          ;先送46h,再送B9h到ISP/IAP触发寄存器,每次都需如此
    MOV   ISP_TRIG,  #0B9h         ;送完B9h后，ISP/IAP命令立即被触发起动
;CPU等待IAP动作完成后，才会继续执行程序.
    NOP                            ;字节编程成功后，CPU继续执行程序
;以下语句可不用,只是出于安全考虑而已
    MOV   ISP_CONTR, #00000000B ;禁止ISP/IAP操作
    MOV   ISP_CMD,   #00000000B ;去除ISP/IAP命令
    MOV   ISP_TRIG,  #00000000B ;防止ISP/IAP命令误触发
    ;MOV  ISP_ADDRH, #0         ;送地址高字节单元为00,指向非EEPROM区,防止误操作
    ;MOV  ISP_ADDRL, #0         ;送地址低字节单元为00,指向非EEPROM区,防止误操作
```

小常识： (STC单片机的Data Flash 当EEPROM功能使用)

3个基本命令----字节读，字节编程，扇区擦除

字节编程：如果该字节是“1111,1111B”，则可将其中的“1”编程为“0”，如果该字节中有位为“0”，则须先将整个扇区擦除，因为只有“扇区擦除”才可以将“0”变为“1”。

扇区擦除：只有“扇区擦除”才可能将“0”擦除为“1”。

大建议：

1.同一次修改的数据放在同一扇区中，不是同一次修改的数据放在另外的扇区,就不须读出保护。

2.如果一个扇区只用一个字节，那就是真正的EEPROM,STC单片机的Data Flash比外部EEPROM要快很多，读一个字节/编程一个字节/擦除一个扇区大概是10uS/60uS/10mS。

3.如果在一个扇区中存放了大量的数据，某次只需要修改其中的一个字节或一部分字节时，则另外的不需要修改的数据须先读出放在STC 单片机的RAM中，然后擦除整个扇区，再将需要保留的数据和需修改的数据一并写回该扇区中。这时每个扇区使用的字节数是使用的越少越方便(不需读出一大堆需保留数据)。

;扇区擦除，没有字节擦除，只有扇区擦除，512 字节 / 扇区,每个扇区用得越少越方便

;如果要对某个扇区进行擦除，而其中有些字节的内容需要保留，则需将其先读到单片机

;内部的 RAM 中保存，再将该扇区擦除，然后将须保留的数据写回该扇区，所以每个扇区

;中用的字节数越少越好，操作起来越灵活越快.

```
    MOV   ISP_ADDRH,#SECTOR_FIRST_BYTE_ADDR_HIGH ;送扇区起始地址高字节 ┐ 地址需要改变时
    MOV   ISP_ADDRL,#SECTOR_FIRST_BYTE_ADDR_LOW  ;送扇区起始地址低字节 ┘ 才需重新送地址
    MOV   ISP_CONTR,#WAIT_TIME ;设置等待时间 ┐
    ORL   ISP_CONTR,#10000000B ;允许 ISP/IAP ┘ 此两句可以合成一句,并且只送一次就够了
    MOV   ISP_CMD,   #ISP_IAP_SECTOR_ERASE;送扇区擦除命令,命令不需改变时,不需重新送命令
    MOV   ISP_TRIG,       #46h       ;先送 46h,再送 B9h 到 ISP/IAP 触发寄存器,每次都需如此
    MOV   ISP_TRIG,       #0B9h;送完 B9h 后，ISP/IAP 命令立即被触发起动
;CPU 等待 IAP 动作完成后，才会继续执行程序.
    NOP                              ;扇区擦除成功后，CPU 继续执行程序
;以下语句可不用,只是出于安全考虑而已
    MOV   ISP_CONTR,#00000000B       ;禁止 ISP/IAP 操作
    MOV   ISP_CMD,   #00000000B      ;去除 ISP/IAP 命令
    MOV   ISP_TRIG,  #00000000B      ;防止 ISP/IAP 命令误触发
    ;MOV  ISP_ADDRH,#0               ;送地址高字节单元为 00,指向非 EEPROM 区
    ;MOV  ISP_ADDRL,#0               ;送地址低字节单元为 00,防止误操作
```

```
;从用户应用程序区(AP区)软件复位并切换到ISP程序区开始执行程序
    MOV   ISP_CONTR,#01100000B ;SWBS = 1(选择 ISP 区), SWRST = 1(软复位)

;从 ISP 程序区软件复位并切换到用户应用程序区(AP 区)开始执行程序
    MOV   ISP_CONTR,#00100000B ;SWBS = 0(选择 AP 区), SWRST = 1(软复位)
```

;使用 ISP/IAP 功能的朋友尽量给 13922805190（姚工）一个电话交流一下

```
;STC12C5410AD 系列和 STC12C2052AD 系列单片机 EEPROM/IAP 功能测试程序演示
;/* ---   STC International Limited ---------------- */
;/* ---   宏晶科技 姚永平  设计 2006/1/6   V1.0 ----- */
;/* ---   演示 STC12C5410AD 系列 MCU EEPROM/IAP 功能 - */
;/* ---   演示 STC12C2052AD 系列 MCU EEPROM/IAP 功能 - */
;/* ---   Mobile: 13922805190 ---------------------- */
;/* ---   Fax: 0755-82944243 ----------------------- */
;/* ---   Tel: 0755-82948409 ----------------------- */
;/* ---   Web: www.mcu-memory.com ------------------ */
;本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过,EEPROM 的数据在 P1 口上显示
;-----------------------------------------------
;声明与 IAP/ISP/EEPROM 有关的特殊功能寄存器的地址
ISP_DATA        EQU   0E2H
ISP_ADDRH       EQU   0E3H
ISP_ADDRL       EQU   0E4H
ISP_CMD         EQU   0E5H
ISP_TRIG        EQU   0E6H
ISP_CONTR       EQU   0E7H
;定义 ISP/IAP 命令
ISP_IAP_BYTE_READ       EQU   1H  ;字节读
ISP_IAP_BYTE_PROGRAM    EQU   2H  ;字节编程,可以将 1 写成 0，要将 1 变成 0，必须执行字节编程
ISP_IAP_SECTOR_ERASE    EQU   3H  ;扇区擦除,可以将 0 擦成 1，要将 0 变成 1，必须擦除整个扇区
;定义 Flash  操作等待时间及允许 IAP/ISP/EEPROM 操作的常数
;ENABLE_ISP       EQU    80H      ;系统工作时钟<30MHz 时，对 ISP_CONTR 寄存器设置此值
;ENABLE_ISP       EQU    81H      ;系统工作时钟<24MHz 时，对 ISP_CONTR 寄存器设置此值
 ENABLE_ISP       EQU    82H      ;系统工作时钟<20MHz 时，对 ISP_CONTR 寄存器设置此值
;ENABLE_ISP       EQU    83H      ;系统工作时钟<12MHz 时，对 ISP_CONTR 寄存器设置此值
;ENABLE_ISP       EQU    84H      ;系统工作时钟<6MHz 时，对 ISP_CONTR 寄存器设置此值
;ENABLE_ISP       EQU    85H      ;系统工作时钟<3MHz 时，对 ISP_CONTR 寄存器设置此值
;ENABLE_ISP       EQU    86H      ;系统工作时钟<2MHz 时，对 ISP_CONTR 寄存器设置此值
;ENABLE_ISP       EQU    87H      ;系统工作时钟<1MHz 时，对 ISP_CONTR 寄存器设置此值

DEBUG_DATA        EQU    5AH  ;是本测试程序选定的 EEPROM 单元的数值如正确应等于的数值
;-----------------------
;选择 MCU 型号
;DATA_FLASH_START_ADDRESS  EQU  1000H  ;STC12C2052AD 系列单片机的 EEPROM 测试起始地址
DATA_FLASH_START_ADDRESS  EQU  2800H   ;STC12C5410AD 系列单片机的 EEPROM 测试起始地址
;-----------------------------------------------
    ORG    0000H
    LJMP   MAIN
;-----------------------------------------------
    ORG    0100H
MAIN:
    MOV     P1,#0F0H        ;演示程序开始工作,让 P1.0/P1.1/P1.2/P1.3 控制的灯亮
    LCALL   Delay           ;延时
    MOV     P1,#0FH         ;演示程序开始工作,让 P1.7/P1.6/P1.5/P1.4 控制的灯亮
```

```
    LCALL   Delay           ;延时
    MOV     SP,  #7FH       ;堆栈指针指向 7FH 单元
;***************************************************
;将 EEPROM 测试起始地址单元的内容读出
MAIN1:
    MOV     DPTR, #DATA_FLASH_START_ADDRESS   ;将 EEPROM 测试起始地址送 DPTR 数据指针
    LCALL   Byte_Read
    MOV     40H, A                      ;将 EEPROM 的值送 40H 单元保存
    CJNE    A, #DEBUG_DATA, DATA_NOT_EQU_DEBUG_DATA     ;如果数据比较不正确,就跳转
DATA_IS_DEBUG_DATA:
;数据是对的,亮 P1.7 控制的灯,然后在 P1 口上将 EEPROM 的数据显示出来
    MOV     P1,  #01111111B   ;如 (DATA_FLASH_START_ADDRESS)的值等于#DEBUG_DATA, 亮 P1.7
    LCALL   Delay                 ;延时
    MOV     A, 40H    ;将保存在 40H 单元中 EEPROM 的值从 40H 单元送累加器 A
    CPL     A           ;取反的目的是相应的灯亮代表 1,不亮代表 0
    MOV     P1,A      ;数据是对的, 送 P1 显示
WAIT1:
    SJMP    WAIT1     ;数据是对的, 送 P1 显示后, CPU 在此无限循环执行此句

DATA_NOT_EQU_DEBUG_DATA:
;EEPROM 里的数据是错的,亮 P1.3 控制的灯,然后在 P1 口上将错误的数据显示出来,
;再将该 EEPROM 所在的扇区整个擦除,将正确的数据写入后,亮 P1.5 控制的灯
    MOV     P1,  #11110111B ;如 (DATA_FLASH_START_ADDRESS)的值不等于#DEBUG_DATA, 亮 P1.3
    LCALL  Delay       ;延时
    MOV     A,    40H   ;将保存在 40H 单元中 EEPROM 的值从 40H 单元送累加器 A
    CPL     A            ;取反的目的是相应的灯亮代表 1,不亮代表 0
    MOV     P1,  A     ;数据不对, 送 P1 显示
    LCALL   Delay      ;延时

    MOV     DPTR,#DATA_FLASH_START_ADDRESS    ;将 EEPROM 测试起始地址送 DPTR 数据指针
    ACALL   Sector_Erase   ;擦除整个扇区
    MOV     DPTR, #DATA_FLASH_START_ADDRESS    ;将 EEPROM 测试起始地址送 DPTR 数据指针
    MOV     A,     #DEBUG_DATA         ;写入 EEPROM 的数据为 #DEBUG_DATA
    ACALL   Byte_Program               ;字节编程
    MOV     P1,    #11011111B          ;将先前亮的 P1.3 灯关闭 ,再亮 P1.5 灯,代表数据已被修改
WAIT2:
    SJMP    WAIT2          ;字节编程后,CPU 在此无限循环执行此句
;***************************************************
```

```
;-----------------------------------------------
;读一字节,调用前需打开 IAP 功能,入口:DPTR = 字节地址,返回:A = 读出字节
Byte_Read:
    MOV     ISP_CONTR,  #ENABLE_ISP       ;打开 IAP 功能, 设置 Flash 操作等待时间
    MOV     ISP_CMD,    #ISP_IAP_BYTE_READ  ;设置为 IAP/ISP/EEPROM 字节读模式命令
    MOV     ISP_ADDRH,  DPH               ;设置目标单元地址的高 8 位地址
    MOV     ISP_ADDRL,  DPL               ;设置目标单元地址的低 8 位地址
    ;CLR     EA
    MOV     ISP_TRIG,   #46H          ;先送 46h,再送 B9h 到 ISP/IAP 触发寄存器,每次都需如此
    MOV     ISP_TRIG,   #0B9H         ;送完 B9h 后, ISP/IAP 命令立即被触发起动
    NOP
    MOV     A,    ISP_DATA            ;读出的数据在 ISP_DATA 单元中,送入累加器 A
    ;SETB    EA
    ACALL IAP_Disable ;关闭 IAP 功能, 清相关的特殊功能寄存器,使 CPU 处于安全状态,
                      ;一次连续的 IAP 操作完成之后建议关闭 IAP 功能,不需要每次都关
    RET
;-----------------------------------------------
;字节编程,调用前需打开 IAP 功能,入口:DPTR = 字节地址, A= 须编程字节的数据
Byte_Program:
    MOV    ISP_CONTR, #ENABLE_ISP     ;打开 IAP 功能, 设置 Flash 操作等待时间
    MOV    ISP_CMD, #ISP_IAP_BYTE_PROGRAM    ;设置为 IAP/ISP/EEPROM 字节编程模式命令
    MOV    ISP_ADDRH, DPH              ;设置目标单元地址的高 8 位地址
    MOV    ISP_ADDRL, DPL              ;设置目标单元地址的低 8 位地址
    MOV    ISP_DATA, A                 ;要编程的数据先送进 ISP_DATA 寄存器
    ;CLR    EA
    MOV    ISP_TRIG, #46H              ;先送 46h,再送 B9h 到 ISP/IAP 触发寄存器,每次都需如此
    MOV    ISP_TRIG, #0B9H             ;送完 B9h 后, ISP/IAP 命令立即被触发起动
    NOP
    ;SETB   EA
     ACALL IAP_Disable ;关闭 IAP 功能, 清相关的特殊功能寄存器,使 CPU 处于安全状态,
                       ;一次连续的 IAP 操作完成之后建议关闭 IAP 功能,不需要每次都关
    RET
;-----------------------------------------------
;擦除扇区, 入口:DPTR = 扇区地址
Sector_Erase:
    MOV    ISP_CONTR, #ENABLE_ISP     ;打开 IAP 功能, 设置 Flash 操作等待时间
    MOV    ISP_CMD, #03H               ;设置为 IAP/ISP/EEPROM 扇区擦除模式命令
    MOV    ISP_ADDRH, DPH              ;设置目标单元地址的高 8 位地址
    MOV    ISP_ADDRL, DPL              ;设置目标单元地址的低 8 位地址
    ;CLR    EA
    MOV    ISP_TRIG, #46H              ;先送 46h,再送 B9h 到 ISP/IAP 触发寄存器,每次都需如此
    MOV    ISP_TRIG, #0B9H             ;送完 B9h 后, ISP/IAP 命令立即被触发起动
    NOP
    ;SETB   EA
     ACALL IAP_Disable ;关闭 IAP 功能, 清相关的特殊功能寄存器,使 CPU 处于安全状态,
                       ;一次连续的 IAP 操作完成之后建议关闭 IAP 功能,不需要每次都关
    RET
```

```
;-----------------------------------------------
IAP_Disable:
;关闭 IAP 功能, 清相关的特殊功能寄存器,使CPU处于安全状态,
;一次连续的IAP操作完成之后建议关闭IAP功能,不需要每次都关
    MOV    ISP_CONTR, #0          ;关闭 IAP 功能
    MOV    ISP_CMD,  #0           ;清命令寄存器,使命令寄存器无命令,此句可不用
    MOV    ISP_TRIG, #0           ;清命令触发寄存器,使命令触发寄存器无触发,此句可不用
    RET
;-----------------------------------------------
Delay:
    CLR    A
    MOV    R0, A
    MOV    R1, A
    MOV    R2, #20H
Delay_Loop:
    DJNZ  R0, Delay_Loop
    DJNZ  R1, Delay_Loop
    DJNZ  R2, Delay_Loop
    RET

;-----------------------------------------------

    END
;***********************************************
```

STC12C5410AD 系列单片机定时器的使用

定时器 0 和 1

定时和计数功能由特殊功能寄存器 TMOD 的控制位 C/$\overline{T}$ 进行选择，TMOD 寄存器的各位信息如下表所列。可以看出，2 个定时 / 计数器有 4 种操作模式，通过 TMOD 的 M1 和 M0 选择。2 个定时 / 计数器的模式 0 、1 和 2 都相同，模式 3 不同，各模式下的功能如下所述。

寄存器 TMOD 各位的功能描述

TMOD　　地址：89H　　复位值：00H

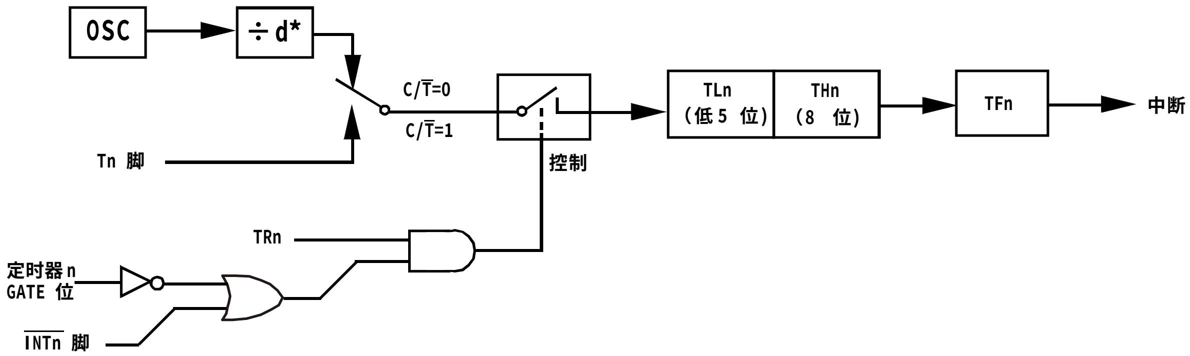
不可位寻址

7	6	5	4	3	2	1	0
GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
定时器 1				定时器 0			

位	符号	功能
TMOD.7/	GATE	TMOD.7 控制定时器 1,置 1 时只有在 $\overline{INT1}$ 脚为高及 TR1 控制位置 1 时才可打开定时器 / 计数器 1。
TMOD.3/	GATE	TMOD.3 控制定时器 0,置 1 时只有在 $\overline{INT0}$ 脚为高及 TR0 控制位置 1 时才可打开定时器 / 计数器 0。
TMOD.6/	C/$\overline{T}$	TMOD.6 控制定时器 1 用作定时器或计数器，清零则用作定时器（从内部系统时钟输入），置 1 用作计数器（从 T1/P3.5 脚输入）
TMOD.2/	C/$\overline{T}$	TMOD.2 控制定时器 0 用作定时器或计数器，清零则用作定时器（从内部系统时钟输入），置 1 用作计数器（从 T0/P3.4 脚输入）
TMOD.5/TMOD.4	M1、M0	定时器 / 计数器 1 模式选择
	0　0	13 位定时器 / 计数器，兼容 8048 定时器模式，TL1 只用低 5 位参与分频，TH1 整个 8 位全用。
	0　1	16 位定时器 / 计数器，TL1、TH1 全用
	1　0	8 位自动重装载定时器，当溢出时将 TH1 存放的值自动重装入 TL1。
	1　1	定时器 / 计数器 1 此时无效（停止计数）。
TMOD.1/TMOD.0	M1、M0	定时器 / 计数器 0 模式选择
	0　0	13 位定时器 / 计数器，兼容 8048 定时器模式，TL0 只用低 5 位参与分频，TH0 整个 8 位全用。
	0　1	16 位定时器 / 计数器，TL0、TH0 全用
	1　0	8 位自动重装载定时器，当溢出时将 TH0 存放的值自动重装入 TL0。
	1　1	定时器 0 此时作为双 8 位定时器 / 计数器。TL0 作为一个 8 位定时器 / 计数器，通过标准定时器 0 的控制位控制。TH0 仅作为一个 8 位定时器，由定时器 1 的控制位控制。

1. 模式 0

将定时器设置成模式 0 时类似 8048 定时器，即 8 位计数器带 32 分频的预分频器。下图所示为模式 0 工作方式。此模式下，定时器配置为 13 位的计数器，由 TLn 的低 5 位和 THn 的 8 位所构成。TLn 低 5 位溢出向 THn 进位，THn 计数溢出置位 TCON 中的溢出标志位 TFn （n=0，1)。GATE=0 时，如 TRn=1，则定时器计数。GATE=1 时，允许由外部输入 $\overline{INT1}$ 控制定时器 1，$\overline{INT0}$ 控制定时器 0，这样可实现脉宽测量。TRn 为 TCON 寄存器内的控制位，TCON 寄存器各位的具体功能描述见 TCON 寄存器各位的具体功能描述表。



* 在T0x12 = 0模式下，d=12(12时钟模式)； 在T0x12 = 1模式下，d=1(1T)。

图 定时器/计数器0和定时器/计数器1的模式0 ： 13位定时/计数器

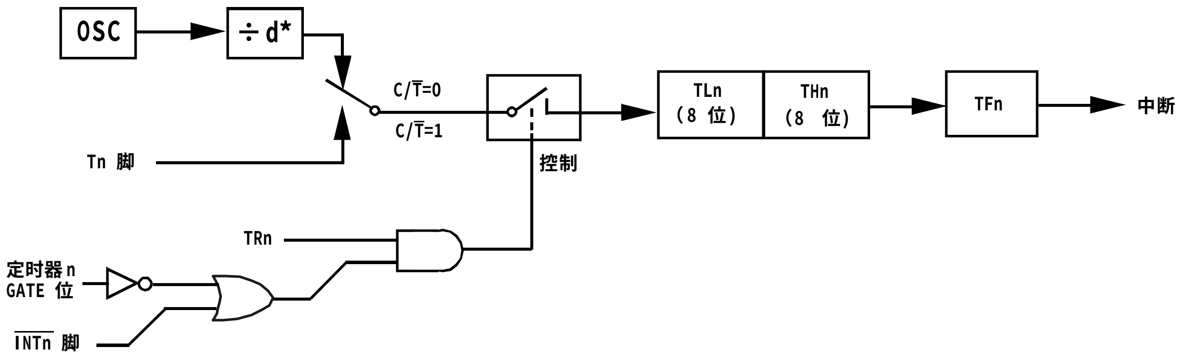
表 寄存器TCON各位的功能描述

TCON 地址：88H									
可位寻址 复位值：00H		7	6	5	4	3	2	1	0
		TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
位	符 号	功 能							
TCON.7	TF1	定时器/计数器1溢出标志位。当T1被允许计数后，T1从初值开始加1计数，最高位产生溢出时，置“1”TF1，并向CPU请求中断，当CPU响应时，由硬件清“0”TF1，TF1也可以由程序查询或清“0”。							
TCON.6	TR1	定时器T1的运行控制位。该位由软件置位和清零。当GATE（TMOD.7）=0，TR1=1时就允许T1开始计数，TR1=0时禁止T1计数。当GATE（TMOD.7）=1，TR1=1且INT1输入高电平时，才允许T1计数。							
TCON.5	TF0	定时器/计数器0溢出标志位。当T0被允许计数后，T0从初值开始加1计数，最高位产生溢出时，置“1”TF0，并向CPU请求中断，当CPU响应时，由硬件清“0”TF0，TF0也可以由程序查询或清“0”。							
TCON.4	TR0	定时器T0的运行控制位。该位由软件置位和清零。当GATE（TMOD.3）=0，TR0=1时就允许T0开始计数，TR1=0时禁止T0计数。当GATE（TMOD.3）=1，TR0=1且$\overline{INT0}$输入高电平时，才允许T0计数。							
TCON.3	IE1	外部中断1中断请求标志位。当主机响应中断转向该中断服务程序执行时，由内部硬件自动将IE1位清0。							
TCON.2	IT1	外部中断1触发方式控制位。IT1=0时，外部中断1为低电平触发方式，当$\overline{INT1}$（P3.3）输入低电平时，置位IE1。采用低电平触发方式时，外部中断源（输入到INT1）必须保持低电平有效，直到该中断被CPU响应，同时在该中断服务程序执行完之前，外部中断源必须被清除（P3.3要变高），否则将产生另一次中断。当IT1=1时，则外部中断1（$\overline{INT1}$）端口由“1”→“0”下降沿跳变，激活中断请求标志位IE1，向主机请求中断处理。							
TCON.1	IE0	外部中断0中断请求标志位。当主机响应中断转向该中断服务程序执行时，由内部硬件自动将IE0位清0。							
TCON.0	IT0	外部中断1触发方式控制位。IT0=0时，外部中断0为低电平触发方式，当$\overline{INT0}$（P3.2）输入低电平时，置位IE0。采用低电平触发方式时，外部中断源（输入到$\overline{INT0}$）必须保持低电平有效，直到该中断被CPU响应，同时在该中断服务程序执行完之前，外部中断源必须被清除（P3.2要变高），否则将产生另一次中断。当IT0=1时，则外部中断0（$\overline{INT0}$）端口由“1”→“0”下降沿跳变，激活中断请求标志位IE1，向主机请求中断处理。							

该13位寄存器包含THn全部8个位及TLn的低5位。TLn的高3位不定，可将其忽略。置位运行标志（TRn）不能清零此寄存器。模式0的操作对于定时器0及定时器1都是相同的。2个不同的GATE位（TMOD.7和TMOD.3）分别分配给定时器1及定时器0。

2．模式 1

模式 1 除了使用了 THn 及 TLn 全部 16 位外，其他与模式 0 完全相同。

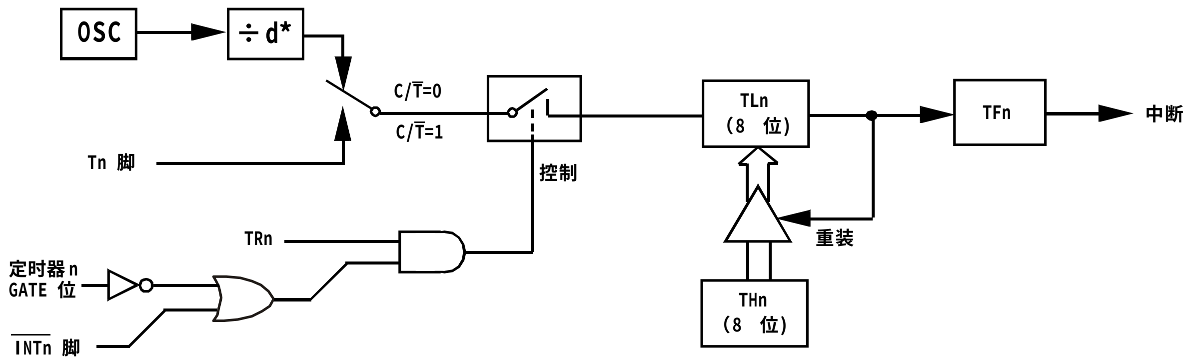


* 在 T0x12 = 0 模式下，d=12(12 时钟模式)； 在 T0x12 = 1 模式下，d=1(1T)。

图 定时器 / 计数器 0 和定时器 / 计数器 1 的模式 1 ： 16 位定时 / 计数器

3．模式 2

此模式下定时器 / 计数器 0 和 1 作为可自动重装载的 8 位计数器（TLn），如下图所示。TLn 的溢出不仅置位 TFn，而且将 THn 内容重新装入 TLn，THn 内容由软件预置，重装时 THn 内容不变。模式 2 的操作对于定时器 0 及定时器 1 是相同的。



* 在 T0x12 = 0 模式下，d=12(12 时钟模式)； 在 T0x12 = 1 模式下，d=1(1T)。

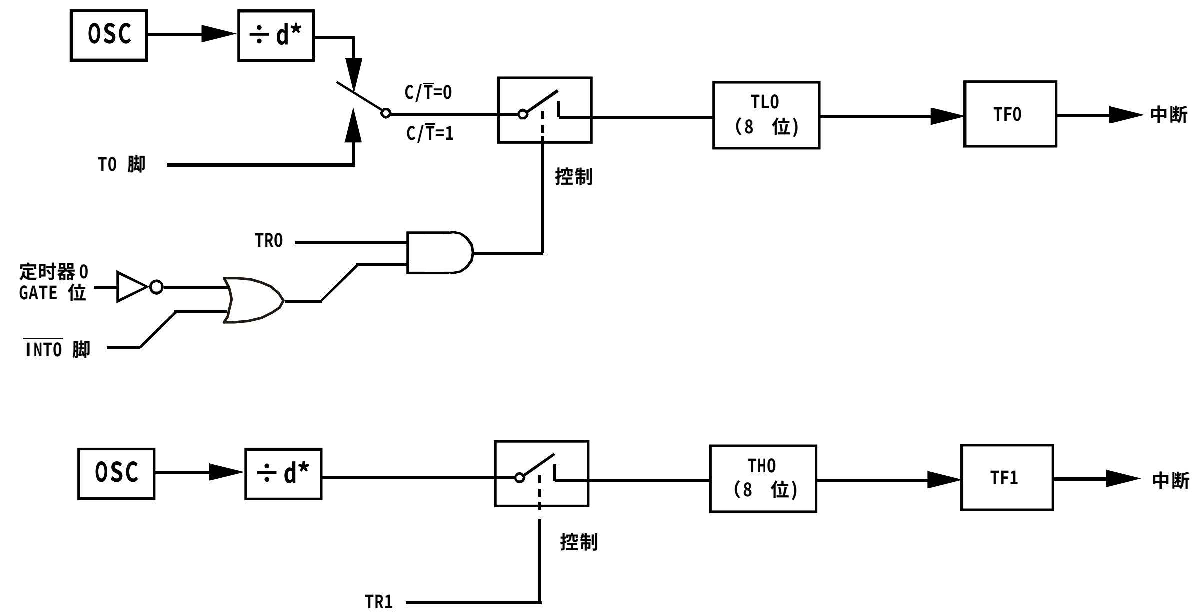
图 定时器 / 计数器 0 和 1 的模式 2 ： 8 位自动重装载

4.模式3

对定时器1，在模式3 时，定时器1 停止计数，效果与将TR1 设置为0 相同。

对定时器0，此模式下定时器0的TL0及TH0作为2个独立的8位计数器。下图为模式3时的定时器0逻辑图。TL0占用定时器0的控制位：C/$\overline{T}$、GATE、TR0、$\overline{INT0}$及TF0。TH0限定为定时器功能（计数器周期），占用定时器1 的TR1 及TF1。此时，TH0 控制定时器1 中断。

模式3 是为了增加一个附加的8 位定时器/计数器而提供的，使单片机具有三个定时器/计数器。模式3只适用于定时器/计数器0，定时器T1处于模式3时相当于TR1=0，停止计数（此时T1可用来作串行口波特率发生器），而T0 可作为两个定时器用。



* 在T0x12 = 0模式下，d=12(12时钟模式)；　在T0x12 = 1模式下，d=1(1T)。

图　定时/计数器0的模式3　：　两个8位计数器

定时器应用举例

【例 1】 定时 / 计数器编程，定时 / 计数器的应用编程主要需考虑：根据应用要求，通过程序初始化，正确设置控制字，正确计算和计算计数初值，编写中断服务程序，适时设置控制位等。通常情况下，设置顺序大致如下：

1) 工作方式控制字（TMOD、T2CON）的设置；

2) 计数初值的计算并装入 THx、TLx、RCAP2H、RCAP2L；

3) 中断允许位 ETx、EA 的设置，使主机开放中断；

4) 启 / 停位 TRx 的设置等。

现以定时 / 计数器 0 或 1 为例作一简要介绍。

8051 系列单片机的定时器 / 计数器 0 或 1 是以不断加 1 进行计数的，即属加 1 计数器，因此，就不能直接将实际的计数值作为计数初值送入计数寄存器 THx、TLx 中去，而必须将实际计数值以 2^8、2^{13}、2^{16} 为模求补，以其补码作为计数初值设置 THx 和 TLx。

设：实际计数值为 X，计数器长度为 n（n=8、13、16），则应装入计数器 THx、TLx 中的计数初值为 2^n-x，式中 2^n 为取模值。例如，工作方式 0 的计数长度为 13 位，则 n=13，以 2^{13} 为模，工作方式 1 的计数长度为 16，则 n=16，以 2^{16} 为模等等。所以，计数初值为（x）= 2^n-x。

对于定时模式，是对机器周期计数，而机器周期与选定的主频密切相关。因此，需根据应用系统所选定的主频计算出机器周期值。现以主频 6MHz 为例，则机器周期为：

$$\text{一个机器周期} = \frac{12}{\text{主振频率}} = \frac{12}{6\times10^6}\mu s = 2\mu s$$

实际定时时间 Tc = x · Tp

式中 Tp 为机器周期，Tc 为所需定时时间，x 为所需计数次数。Tp 和 Tp 一般为已知值，在求出 Tp 后即可求得所需计数值 x，再将 x 求补码，即求得定时计数初值。即

$$(x)\text{补} = 2^n - x$$

例如，设定时时间 Tc = 5ms，机器周期 TP = 2μs，可求得定时计数次数

$$x = \frac{5ms}{2\mu s} = 2500\text{次}$$

设选用工作方式 1，则 n=16，则应设置的定时时间计数初值为：(x) 补 = 2^{16} - x=65536-2500=63036，还需将它分解成两个 8 位十六进制数，分别求得低 8 位为 3CH 装入 TLx，高 8 位为 F6H 装入 THx 中。

工作方式 0、1、2 的最大计数次数分别为 8192、65536 和 256。

对外部事件计数模式，只需根据实际计数次数求补后变换成两个十六进制码即可。

【例 2】 定时 / 计数器应用编程，设某应用系统，选择定时 / 计数器 1 定时模式，定时时间 Tc = 10ms，主频频率为 12MHz，每 10ms 向主机请求处理。选定工作方式 1。计算得计数初值：低 8 位初值为 F0H，高 8 位初值为 D8H。

（1）初始化程序

所谓初始化，一般在主程序中根据应用要求对定时 / 计数器进行功能选择及参数设定等预置程序，本例初始化程序如下：

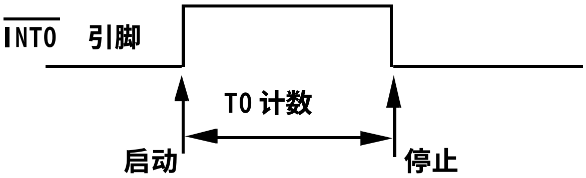
```
START：
          ⋮                    ；主程序段
        MOV   SP，#60H         ；设置堆栈区域
        MOV   TMOD，#10H       ；选择 T 1 、定时模式，工作方式 1
        MOV   TH1，#0D8H       ；设置高字节计数初值
        MOV   TL1，#0F0H       ；设置低字节计数初值
        SETB  EA               }
        SETB  ET1              }  开中断
          ⋮                    ；其他初始化程序
        SETB  TR1              ；启动 T1 开始计时
          ⋮                    ；继续主程序
```

（2 ）中断服务程序

```
INTT1：  PUSH A                 }
        PUSH DPL               }  现场保护
        PUSH DPH               }
          ⋮
        MOV   TL1,#0F0H        }
        MOV   TH1,#0D8H        }  重新置初值
          ⋮                    ；中断处理主体程序
        POP   DPH              }
        POP   DPL              }  现场恢复
        POP   A                }
        RETI                   ；返回
```

这里展示了中断服务子程序的基本格式。8052 系列单片机的中断属于矢量中断，每一个矢量中断源只留有 8 个字节单元，一般是不够用的，常需用转移指令转到真正的中断服务子程序区去执行。

【例 3】 对外部正脉冲测宽。选择定时 / 计数器 2 进行脉宽测试较方便，但也可选用定时 / 计数器 0 或定时 / 计数器 1 进行测宽操作。本例选用定时 / 计数器 0 （T0）以定时模式，工作方式 1 对 $\overline{\text{INT0}}$ 引脚上的正脉冲进行脉宽测试。



设置 GATE 为 1，机器周期 TP 为 1 μ s。本例程序段编制如下：

```
INTT0：  MOV   TMOD，#09H       ；设 T0 为定时方式 1，GATE 为 1
```

```
            MOV   TL0, #00H          ;}
            MOV   TH0, #00H          ;} TH0,TL0 清0
            CLR   EX0                ;关 INT0 中断
LOP1：      JB    P3.2, LOP1         ;等待 INT0 引低电平
LOP2：      JNB   P3.2, LOP2         ;等待 INT0 引脚高电平
            SETB  TR0                ;启动 T0 开始计数
LOP3：      JB    P3.2, LOP3         ;等待 INT0 低电平
            CLR   TR0                ;停止 T0 计数
            MOV   A, TL0             ;低字节计数值送 A
            MOV   B, TH0             ;高字节计数值送 B
             ⋮                       ;计算脉宽和处理
```

【例 4】 利用定时 / 计数器 0 或定时 / 计数器 1 的 Tx 端口改造成外部中断源输入端口的应用设计。

在某些应用系统中常会出现原有的两个外部中断源 INT0 和 INT1 不够用，而定时 / 计数器有多余，则可将 Tx 用于增加的外部中断源。现选择定时 / 计数器 1 为对外部事件计数模式工作方式 2（自动再装入），设置计数初值为 FFH，则 T1 端口输入一个负跳变脉冲，计数器即回 0 溢出，置位对应的中断请求标志位 TF1 为 1，向主机请求中断处理，从而达到了增加一个外部中断源的目的。应用定时 / 计数器 1（T1）的中断矢量转入中断服务程序处理。其程序示例如下：

（1）主程序段：

```
      ORG   0000H
      AJMP  MAIN             ;转主程序
      ORG   001BH
      LJMP  INTER            ;转 T1 中断服务程序
       ⋮
      ORG   0100             ;主程序入口
MAIN：…
       ⋮
      MOV   SP, #60H         ;设置堆栈区
      MOV   TMOD, #60H       ;设置定时 / 计数器 1，计数方式 2
      MOV   TL1, #0FFH       ;设置计数常数
      MOV   TH1, #0FFH
      SETB  EA               ;开中断
      SETB  ET1              ;开定时 / 计数器 1 中断
      SETB  TR1              ;启动定时 / 计数器 1 计数
       ⋮
```

（2）中断服务程序（具体处理程序略）

```
            ORG   1000H
INTER：     PUSH  A          ;}
            PUSH  DPL        ;} 现场入栈保护
            PUSH  DPH        ;}
             ⋮
```

```
        ⋮                     } 中断处理主体程序
        ⋮
      POP   DPH          ;  }
      POP   DPL          ;  } 现场出栈复原
      POP   A            ;  }
      RETI               ；返回
```

这是中断服务程序的基本格式。

【例5】 某应用系统需通过P1.0和P1.1分别输出周期为200 μs和400 μs的方波。为此，系统选用定时器/计数器0（T0），定时方式3，主频为6MHz，TP=2 μs，经计算得定时常数为9CH和38H。

本例程序段编制如下：

（1）初始化程序段

```
          ⋮
PLT0:MOV   TMOD,#03H        ;设置T0定时方式3
     MOV   TL0, #9CH        ;设置TL0初值
     MOV   TH0, #38H        ;设置TH0初值
     SETB  EA               ;  }
     SETB  ET0              ;  } 开中断
     SETB  ET1              ;  }
     SETB  TR0              ；启动
     SETB  TR1              ；启动
          ⋮
```

(2)中断服务程序段

1)

```
INT0P：  ⋮
         ⋮
     MOV   TL0, #9CH        ;重新设置初值
     CPL   P1.0             ;对P1.0输出信号取反
         ⋮
     RETI                   ；返回
```

2)

```
INT1P    ⋮
         ⋮
     MOV   TH0, #38H        ;重新设置初值
     CPL   P1.1             ;对P1.1输出信号取反
         ⋮
     RETI                   ；返回
```

在实际应用中应注意的问题如下。

（1）定时 / 计数器的实时性

定时 / 计数器启动计数后，当计满回 0 溢出向主机请求中断处理，由内部硬件自动进行。但从回 0 溢出请求中断到主机响应中断并作出处理存在时间延迟，且这种延时随中断请求时的现场环境的不同而不同，一般需延时 3 个机器周期以上，这就给实时处理带来误差。大多数应用场合可忽略不计，但对某些要求实时性苛刻的场合，应采用补偿措施。

这种由中断响应引起的时间延时，对定时 / 计数器工作于方式 0 或 1 而言有两种含义：一是由于中断响应延时而引起的实时处理的误差；二是如需多次且连续不间断地定时 / 计数，由于中断响应延时，则在中断服务程序中再置计数初值时已延误了若干个计数值而引起误差，特别是用于定时就更明显。

例如选用定时方式 1 设置系统时钟，由于上述原因就会产生实时误差。这种场合应采用动态补偿办法以减少系统始终误差。所谓动态补偿，即在中断服务程序中对 THx、TLx 重新置计数初值时，应将 THx、TLx 从回 0 溢出又重新从 0 开始继续计数的值读出，并补偿到原计数初值中去进行重新设置。可考虑如下补偿方法：

```
      ⋮
CLR   EA            ；禁止中断
MOV   A，TLx        ；读 TLx 中已计数值
ADD   A，#LOW       ；LOW 为原低字节计数初值
MOV   TLx，A        ；设置低字节计数初值
MOV   A，#HIGH      ；原高字节计数初值送 A
ADDC  A，THx        ；高字节计数初值补偿
MOV   THx，A        ；置高字节计数初值
SETB  EA            ；开中断
      ⋮
```

（2）动态读取运行中的计数值

在动态读取运行中的定时 / 计数器的计数值时，如果不加注意，就可能出错。这是因为不可能在同一时刻同时读取 THx 和 TLx 中的计数值。比如，先读 TLx 后读 THx，因为定时 / 计数器处于运行状态，在读 TLx 时尚未产生向 THx 进位，而在读 THx 前已产生进位，这时读得的 THx 就不对了；同样，先读 THx 后读 TLx 也可能出错。

一种可避免读错的方法是：先读 THx，后读 TLx，将两次读得的 THx 进行比较；若两次读得的值相等，则可确定读的值是正确的，否则重复上述过程，重复读得的值一般不会再错。此法的软件编程如下：

```
RDTM： MOV  A，THx            ；读取 THx 存 A 中
       MOV  R0，TLx           ；读取 TLx 存 R0 中
       CJNE A，THx，RDTM      ；比较两次 THx 值，若相等，则读得的值正
                              ；确，程序往下执行，否则重读
       MOV  R1，A             ；将 THx 存于 R1 中
        ⋮
```

STC12C5410 系列单片机 PWM/PCA 应用

STC12C5410AD 系列 1T 8051 单片机 PCA/PWM 特殊功能寄存器 PCA/PWM SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
CCON	D8h	PCA Control Register	CF	CR	-	-	CCF3	CCF2	CCF1	CCF0	00xx,0000
CMOD	D9h	PCA Mode Register	CIDL	-	-	-	-	CPS1	CPS0	ECF	0xxx,x000
CCAPM0	DAh	PCA Module 0 Mode Register	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x000,0000
CCAPM1	DBh	PCA Module 1 Mode Register	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x000,0000
CCAPM2	DCh	PCA Module 2 Mode Register	-	ECOM2	CAPP2	CAPN2	MAT2	TOG2	PWM2	ECCF2	x000,0000
CCAPM3	DDh	PCA Module 3 Mode Register	-	ECOM3	CAPP3	CAPN3	MAT3	TOG3	PWM3	ECCF3	x000,0000
CL	E9h	PCA Base Timer Low									0000,0000
CH	F9h	PCA Base Timer High									0000,0000
CCAP0L	EAh	PCA Module-0 Capture Register Low									0000,0000
CCAP0H	FAh	PCA Module-0 Capture Register High									0000,0000
CCAP1L	EBh	PCA Module-1 Capture Register Low									0000,0000
CCAP1H	FBh	PCA Module-1 Capture Register High									0000,0000
CCAP2L	ECh	PCA Module-2 Capture Register Low									0000,0000
CCAP2H	FCh	PCA Module-2 Capture Register High									0000,0000
CCAP3L	EDh	PCA Module-3 Capture Register Low									0000,0000
CCAP3H	FDh	PCA Module-3 Capture Register High									0000,0000
PCA_PWM0	F2h	PCA PWM Mode Auxiliary Register 0	-	-	-	-	-	-	EPC0H	EPC0L	xxxx,xx00
PCA_PWM1	F3h	PCA PWM Mode Auxiliary Register 1	-	-	-	-	-	-	EPC1H	EPC1L	xxxx,xx00
PCA_PWM2	F4h	PCA PWM Mode Auxiliary Register 2	-	-	-	-	-	-	EPC2H	EPC2L	xxxx,xx00
PCA_PWM3	F5h	PCA PWM Mode Auxiliary Register 3	-	-	-	-	-	-	EPC3H	EPC3L	xxxx,xx00

CMOD - PCA 模式 寄存器的位分配 （地址：D9H）

位	7	6	5	4	3	2	1	0
符 号	C I D L	-	-	-	-	C P S 1	C P S 0	E C F

CMOD - PCA 模式 寄存器的位描述 （地址：D9H）

位	符号	描述
7	CIDL	计数器阵列空闲控制：CIDL=0时，空闲模式下PCA计数器继续工作。CIDL＝1时，空闲模式下PCA计数器停止工作。
6 - 3	-	保留为将来之用。
2 - 1	CPS1,CPS0	PCA计数脉冲选择（见下表）。
0	ECF	PCA计数溢出中断使能：ECF=1时，使能寄存器CCON CF位的中断。ECF=0时，禁止该功能。

CMOD - PCA 计数器阵列的计数脉冲选择 （地址：D9H）

CPS1	CPS0	选择PCA时钟源输入
0	0	0，内部时钟，Fosc/12
0	1	1，内部时钟，Fosc/2
1	0	2，定时器0溢出，由于定时器0可以工作在1T方式，所以可以达到计一个时钟就溢出，频率反而是最高的，可达到Fosc
1	1	3，ECI/P3.4脚的外部时钟输入（最大速率＝Fosc/2）

CCON - PCA 控制寄存器的位分配 （地址：D8H）

位	7	6	5	4	3	2	1	0
符 号	CF	CR	-	-	-	-	CCF1	CCF0

CCON - PCA 控制寄存器的位描述 （地址：D8H）

位	符号	描述
7	CF	PCA计数器阵列溢出标志。计数值翻转时该位由硬件置位。如果CMOD寄存器的ECF位置位，CF标志可用来产生中断。CF位可通过硬件或软件置位，但只可通过软件清零。
6	CR	PCA计数器阵列运行控制位。该位通过软件置位，用来起动PCA计数器阵列计数。该位通过软件清零，用来关闭PCA计数器。
5 - 2	-	保留位，保留为将来使用。
1	CCF1	PCA模块1中断标志。当出现匹配或捕获时该位由硬件置位。该位必须通过软件清零。
0	CCF0	PCA模块0中断标志。当出现匹配或捕获时该位由硬件置位。该位必须通过软件清零。

CCAPMn - PCA 比较 / 捕获模块寄存器的位分配（CCAPM0 地址：0DAH；CCAPM1 地址：0DBH）

位	7	6	5	4	3	2	1	0
符 号	-	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn

CCAPMn - PCA 比较 / 捕获模块寄存器的位描述 （ n：0,1,2,3 ）

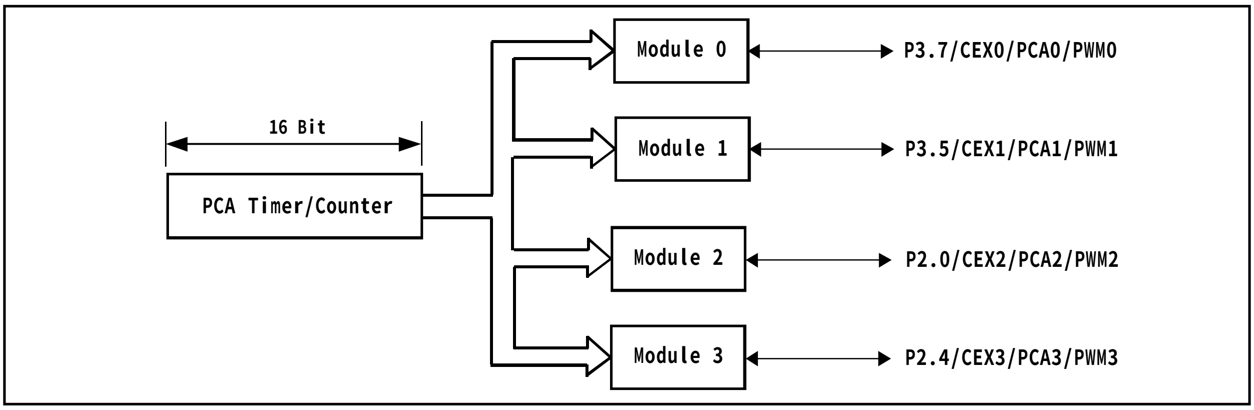
位	符号	描述 n：0,1,2,3
7	-	保留为将来之用。
6	ECOMn	使能比较器。ECOMn＝1时使能比较器功能。
5	CAPPn	正捕获。CAPPn＝1时使能上升沿捕获。
4	CAPNn	负捕获。CAPNn＝1时使能下降沿捕获。
3	MATn	匹配。当MATn＝1时，PCA计数值与模块的比较/捕获寄存器的值的匹配将置位CCON寄存器的中断标志位CCFn。
2	TOGn	翻转。当TOGn＝1时，工作在PCA高速输出模式，PCA计数器的值与模块的比较/捕获寄存器的值的匹配将使CEXn脚翻转。(CEX0/P3.7,CEX1/P3.5,CEX2/P2.0,CEX3/P2.4)
1	PWMn	脉宽调节模式。当PWMn＝1时，使能CEXn脚用作脉宽调节输出。
0	ECCFn	使能CCFn中断。使能寄存器CCON的比较/捕获标志CCFn，用来产生中断。

PCA 模块工作模式（CCAPMn 寄存器，n：0,1,2,3）

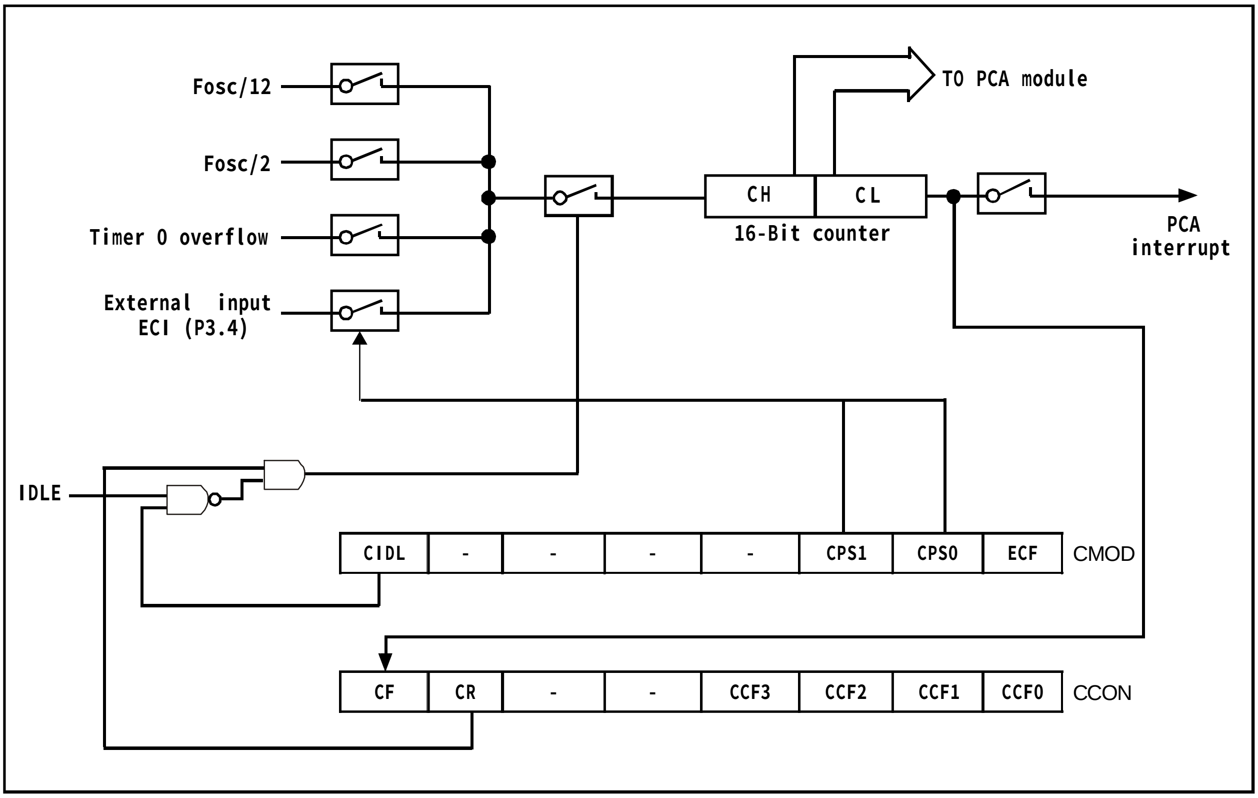
ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn	模块功能
0	0	0	0	0	0	0	无此操作
X	1	0	0	0	0	X	16位捕获模式，由CEXn的上升沿触发
X	0	1	0	0	0	X	16位捕获模式，由CEXn的下降沿触发
X	1	1	0	0	0	X	16位捕获模式，由CEXn的跳变触发
1	0	0	1	0	0	X	16位软件定时器
1	0	0	1	1	0	X	16位高速输出
1	0	0	0	0	1	0	8位PWM

STC12C5410AD 系列单片机可编程计数器阵列（PCA）

PCA 含有一个特殊的 16 位定时器，有 4 个 16 位的捕获 / 比较模块与之相连。每个模块可编程工作在 4 种模式下：上升 / 下降沿捕获、软件定时器、高速输出或可调制脉冲输出。模块 0 连接到 P3.7（CEX0/PCA0/PWM0），模块 1 连接到 P3.5（CEX1/PCA1/PWM1)。模块 2 连接到 P2.0（CEX2/PCA2/PWM2)，模块 3 连接到 P2.4（CEX3/PCA3/PWM3).寄存器 CH 和 CL 的内容是正在自由递增计数的 16 位 PCA 定时器的值。PCA 定时器是 4 个模块的公共时间基准，可通过编程工作在：1/12 振荡频率、1/2 振荡频率、定时器 0 溢出或 ECI 脚的输入（P3.4)。定时器的计数源由 CMOD SFR 的 CPS1 和 CPS0 位来确定（见 CMOD 特殊功能寄存器说明)。



Programmable Counter Array



PCA Timer/Counter

CMOD SFR 还有 2 个位与 PCA 相关。它们分别是：CIDL，空闲模式下允许停止 PCA；ECF，置位时，使能 PCA 中断，当 PCA 定时器溢出将 PCA 计数溢出标志 CF（CCON SFR）置位。

CCON SFR 包含 PCA 的运行控制位（CR）和 PCA 定时器标志（CF）以及各个模块的标志（CCF3/CCF2/CCF1/CCF0)。通过软件置位 CR 位（CCON.6）来运行 PCA。CR 位被清零时 PCA 关闭。当 PCA 计数器溢出时，CF 位（CCON.7）置位,如果 CMOD 寄存器的 ECF 位置位,就产生中断。CF 位只可通过软件清除。CCON 寄存器的位 0～3 是 PCA 各个模块的标志（位 0 对应模块 0，位 1 对应模块 1,位 2 对应模块 2，位 3 对应模块 3），当发生匹配或比较时由硬件置位。这些标志也只能通过软件清除。所有模块共用一个中断向量。PCA 的中断系统如图所示。

PCA 的每个模块都对应一个特殊功能寄存器。它们分别是：模块 0 对应 CCAPM0，模块 1 对应 CCAPM1，模块 2 对应 CCAPM2，模块 3 对应 CCAPM3.特殊功能寄存器包含了相应模块的工作模式控制位。

当模块发生匹配或比较时，ECCFn 位（CCAPMn.0，n ＝0，1，2,3 由工作的模块决定）使能 CCON SFR 的 CCFn 标志来产生中断。

PWM（CCAPMn.1）用来使能脉宽调制模式。

当 PCA 计数值与模块的捕获 / 比较寄存器的值相匹配时，如果 TOG 位（CCAPMn.2）置位，模块的 CEXn 输出将发生翻转。

当 PCA 计数值与模块的捕获 / 比较寄存器的值相匹配时，如果匹配位 MATn（CCAPMn.3）置位，CCON 寄存器的 CCFn 位将被置位。

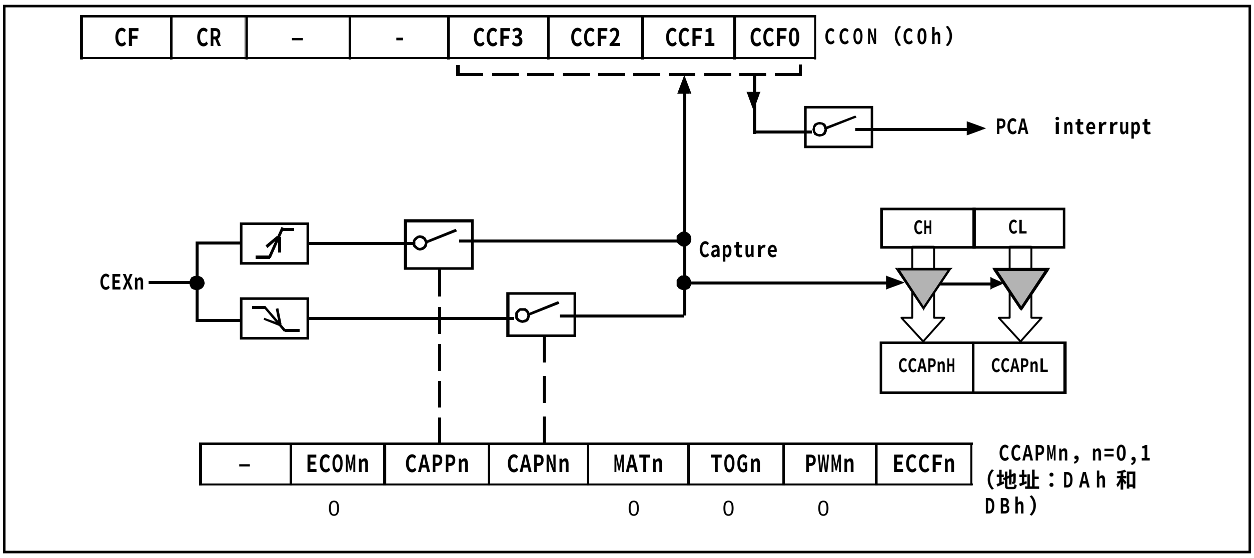
CAPNn（CCAPMn.4）和 CAPPn（CCAPMn.5）用来设置捕获输入的有效沿。CAPNn 位使能下降沿有效，CAPPn 位使能上升沿有效。如果两位都置位，则两种跳变沿都被使能，捕获可在两种跳变沿产 生 。

通过置位 CCAPMn 寄存器的 ECOMn 位（CCAPMn.6）来使能比较器功能。

每个 PCA 模块还对应另外两个寄存器，CCAPnH 和 CCAPnL。当出现捕获或比较时，它们用来保存 16 位的计数值。当 PCA 模块用在 PWM 模式中时，它们用来控制输出的占空比。

PCA 捕获模式

要使一个 PCA 模块工作在捕获模式（下图），寄存器 CCAPMn 的两位（CAPNn 和 CAPPn）或其中任何一位必须置 1。对模块的外部 CEXn 输入（CEX0/P3.7，CEX1/P3.5,CEX2/P2.0,CEX3/P2.4 口）的跳变进行采样。当采样到有效跳变时，PCA 硬件就将 PCA 计数器阵列寄存器（CH 和 CL）的值装载到模块的捕获寄存器中（CCAPnL 和 CCAPnH）。

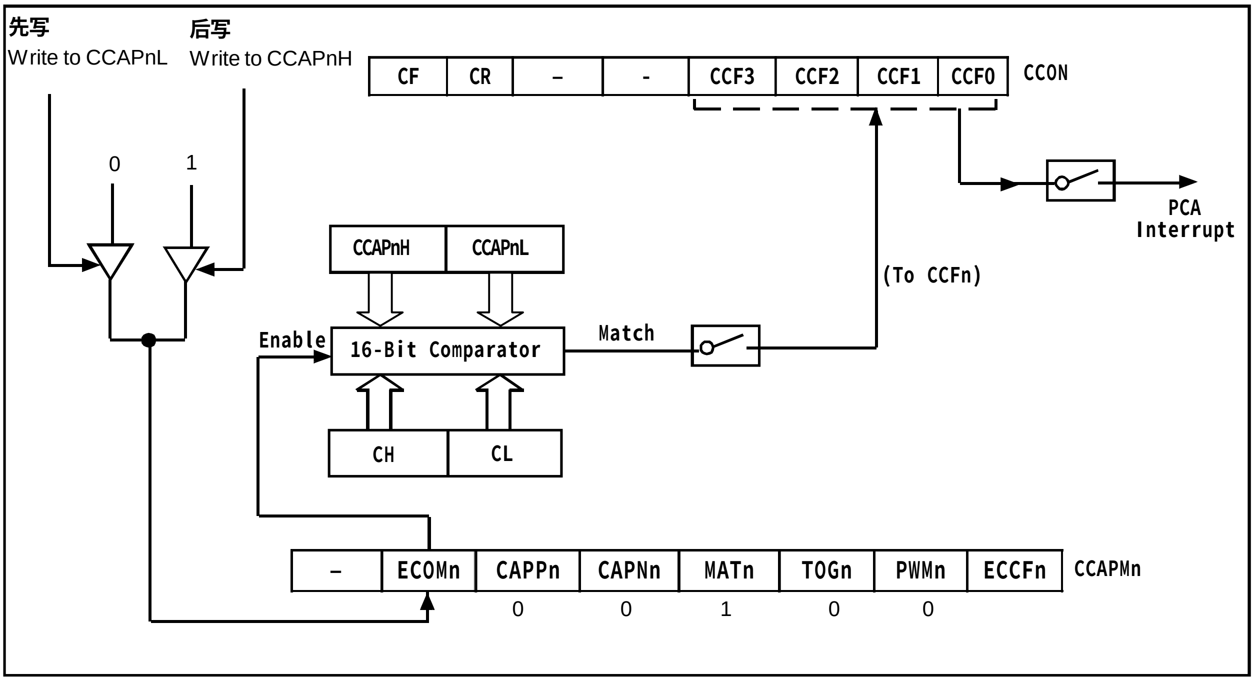


PCA Capture Mode （PCA 捕获模式图）

如果 CCON SFR 的位 CCFn 和 CCAPMn SFR 的位 ECCFn 位被置位，将产生中断。

16 位软件定时器模式

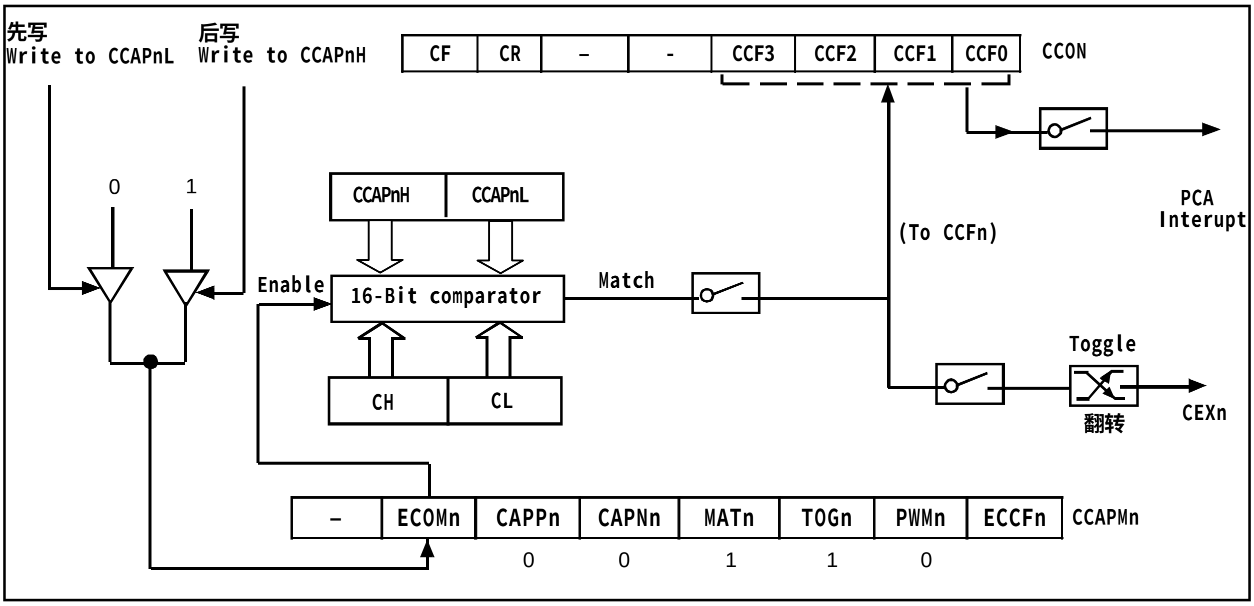
通过置位 CCAPMn 寄存器的 ECOM 和 MAT 位，可使 PCA 模块用作软件定时器（下图）。PCA 定时器的值与模块捕获寄存器的值相比较，当两者相等时，如果位 CCFn（在 CCON SFR 中）和位 ECCFn（在 CCAPMn SFR 中）都置位，将产生中断。



PCA Software Timer Mode/ 软件定时器模式 /PCA 比较模式

高速输出模式

该模式中（下图），当PCA计数器的计数值与模块捕获寄存器的值相匹配时，PCA模块的CEXn输出将发生翻转。要激活高速输出模式，模块CCAPMn SFR的TOG,MAT和ECOM位必须都置位。



PCA High-Speed Output Mode / PCA高速输出模式

在使用PCA高速输出模式时的特别应用注意事项:

如果某一PCA模块工作在高速脉冲输出模式,要用软件输出改变同一组其它普通I/O口的状态，需先做判断CH是否等于CCAPnH，若不等，可自由修改，若相等，再判断CL>CCAPnL情况下才允许改变同一组其它普通I/O口的状态。如用P3.7/PCA0/PWM0做PCA高速脉冲输出，同时程序里面又要用软件输出改变P3.4口的状态时，就需要做判断。

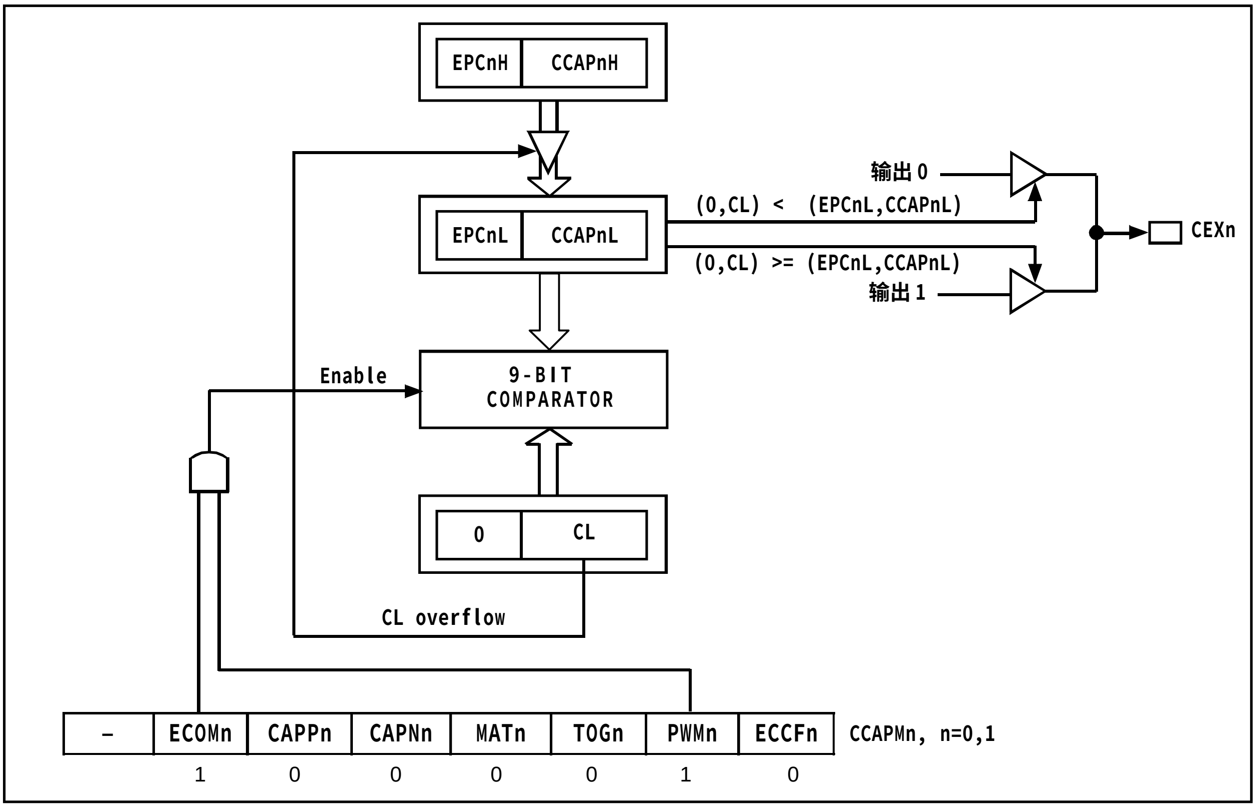
当某个具有PCA高速脉冲输出功能的I/O口工作在高速脉冲输出模式时，如果软件对同一组的其它I/O口进行操作，如果遇上PCA比较器匹配时,该操作有可能会改变此具有PCA高速脉冲输出功能的I/O口的状态，所以同一组的其它I/O口建议不要做输出用，如果做输出用时，要进行判断。

具有PCA高速脉冲输出模式的I/O口	同一组的其它I/O口
PCA0/P3.7	P3.0,P3.1,P3.2,P3.3,P3.4,P3.5
PCA1/P3.5	P3.0,P3.1,P3.2,P3.3,P3.4,P3.7
PCA2/P2.0	P2.1,P2.2,P2.3,P2.4,P2.5,P2.6,P2.7
PCA3/P2.4	P2.0,P2.1,P2.2,P2.3,P2.5,P2.6,P2.7
当以上管脚用在PCA高速脉冲输出模式时，建议同一组的其它I/O口工作在输入模式，如工作在PCA高速输出模式，而同一组的它I/O口又必须工作在输出模式，建议如右列所示。	要用软件输出改变同一组其它普通I/O口的状态，需先做判断CH是否等于CCAPnH，若不等，可自由修改，若相等，再判断CL>CCAPnL情况下才允许改变同一组其它普通I/O口的状态。防止在PCA比较器匹配时做输出工作。

其它几种工作模式无问题

脉宽调节模式

所有 PCA 模块都可用作 PWM 输出（下图）。输出频率取决于 PCA 定时器的时钟源。



PCA PWM mode / 可调制脉冲宽度输出模式

由于所有模块共用仅有的 PCA 定时器，所有它们的输出频率相同。各个模块的输出占空比是独立变化的，与使用的捕获寄存器 {EPCnL，CCAPnL} 有关。当 CL SFR 的值小于 {EPCnL，CCAPnL} 时，输出为低，当 PCA CL SFR 的值等于或大于 {EPCnL，CCAPnL} 时，输出为高。当 CL 的值由 FF 变为 00 溢出时，{EPCnH，CCAPnH} 的内容装载到 {EPCnL，CCAPnL} 中。这样就可实现无干扰地更新 PWM。要使能 PWM 模式，模块 CCAPMn 寄存器的 PWMn 和 ECOMn 位必须置位。

PCA 功能扩展外部中断的示例程序

```
;/* --- STC International Limited ---------------------- */
;/* --- 宏晶科技      姚永平     2006/1/6    V1.0 ---------- */
;/* --- PCA_5410_ASM_INT ------------------------------ */
;/* --- STC12C5412AD, STC12C5410AD, STC12C5408AD ------- */
;/* --- STC12C5406AD, STC12C5404AD, STC12C5402AD ------- */
;/* --- STC12C5052AD, STC12C4052AD, STC12C3052AD ------- */
;/* --- STC12C2052AD, STC12C1052AD, STC12C0552AD ------- */
;/* --- Mobile: 13922805190 --------------------------- */
;/* --- Fax: 0755-82944243 ---------------------------- */
;/* --- Tel: 0755-82948409 ---------------------------- */
;/* --- Web: www.mcu-memory.com ----------------------- */

;-----------------------------------------------------------------------------
;使用 STC12C2052AD 系列单片机 PCA 功能扩展外部中断的示例程序。
;使用 STC12C5410AD 系列单片机 PCA 功能扩展外部中断的示例程序。
;
;P3.7(PCA 模块0) 扩展为下降沿外部中断,
;P3.5(PCA 模块1) 扩展为上升沿/下降沿都可触发的外部中断。
;
;1) 汇编源程序, 把汇编程序产生的的程序代码下载到单片机中, 上电运行本程序。
;2) 将 P3.7/PCA0 短路到地, 这一动作产生一个下降沿,此时本演示程序对 P1.6 取反,
;   P1.6 控制的 LED 灯将会变化一次。
;3) 改变 P3.5/PCA1 的外部高低状态(由高到低--产生下降沿; 由低到高--产生上升沿),
;   本演示程序在 P3.5/PCA1 的下降沿/上升沿都产生中断,此时本演示程序对 P1.5 取反,
;   P1.5 控制的 LED 灯状态将会发生变化。
;   所谓 LED 灯状态发生变化是指 LED 由灭变亮或由亮变灭。
;-----------------------------------------------------------------------------
;声明 STC12C2052AD和STC12C5410AD系列 MCU 特殊功能寄存器地址
IPH       EQU    0B7H              ;中断优先级高位寄存器

EPCA_LVD  EQU    IE.6              ;PCA 中断和 LVD(低压检测)中断共享的总中断控制位
CH        EQU    0F9H              ;PCA 计数器高8位。
CL        EQU    0E9H              ;PCA 计数器低8位。
;-------------------------
CCON      EQU    0D8H              ;PCA 控制寄存器。
CCF0      EQU    CCON.0            ;PCA 模块0 中断标志, 由硬件置位, 必须由软件清0。
CCF1      EQU    CCON.1            ;PCA 模块1 中断标志, 由硬件置位, 必须由软件清0。
CCF2      EQU    CCON.2            ;PCA 模块2 中断标志, 由硬件置位, 必须由软件清0。
CCF3      EQU    CCON.3            ;PCA 模块3 中断标志, 由硬件置位, 必须由软件清0。
CCF4      EQU    CCON.4            ;PCA 模块4 中断标志, 由硬件置位, 必须由软件清0。
CCF5      EQU    CCON.5            ;PCA 模块5 中断标志, 由硬件置位, 必须由软件清0。

CR        EQU    CCON.6            ;1:允许 PCA 计数器计数, 必须由软件清0。
CF        EQU    CCON.7            ;PCA计数器溢出(CH,CL由 FFFFH 变为 0000H)标志,
                                   ;PCA计数器溢出后由硬件置位,必须由软件清0。
```

```
;-----------------------
CMOD      EQU    0D9H              ;PCA 工作模式寄存器。
;CMOD.7      CIDL: idle 状态时 PCA 计数器是否继续计数, 0: 继续计数, 1: 停止计数。

;CMOD.2      CPS1: PCA 计数器计数脉冲源选择位 1。
;CMOD.1      CPS0: PCA 计数器计数脉冲源选择位 0。
;            CPS1   CPS0
;             0       0     外部晶体频率 /12。
;             0       1     外部晶体频率 /2。
;             1       0     Timer 0 溢出脉冲,
;                           Timer 0 还可通过 AUXR 寄存器设置成工作在 12T 或 1T 模式。
;             1       1     从 ECI/P3.4 脚输入的外部时钟。

;CMOD.0      ECF: PCA 计数器溢出中断允许位, 1-- 允许 CF(CCON.7) 产生中断。
;-----------------------

CCAP0H    EQU    0FAH              ;PCA 模块 0 的捕捉 / 比较寄存器高 8 位。
CCAP1H    EQU    0FBH              ;PCA 模块 1 的捕捉 / 比较寄存器高 8 位。
CCAP2H    EQU    0FCH              ;PCA 模块 2 的捕捉 / 比较寄存器高 8 位。
CCAP3H    EQU    0FDH              ;PCA 模块 3 的捕捉 / 比较寄存器高 8 位。
CCAP4H    EQU    0FEH              ;PCA 模块 4 的捕捉 / 比较寄存器高 8 位。
CCAP5H    EQU    0FFH              ;PCA 模块 5 的捕捉 / 比较寄存器高 8 位。

CCAP0L    EQU    0EAH              ;PCA 模块 0 的捕捉 / 比较寄存器低 8 位。
CCAP1L    EQU    0EBH              ;PCA 模块 1 的捕捉 / 比较寄存器低 8 位。
CCAP2L    EQU    0ECH              ;PCA 模块 2 的捕捉 / 比较寄存器低 8 位。
CCAP3L    EQU    0EDH              ;PCA 模块 3 的捕捉 / 比较寄存器低 8 位。
CCAP4L    EQU    0EEH              ;PCA 模块 4 的捕捉 / 比较寄存器低 8 位。
CCAP5L    EQU    0EFH              ;PCA 模块 5 的捕捉 / 比较寄存器低 8 位。

;-----------------------
PCA_PWM0 EQU    0F2H              ;PCA 模块 0 PWM 寄存器。
PCA_PWM1 EQU    0F3H              ;PCA 模块 1 PWM 寄存器。
PCA_PWM2 EQU    0F4H              ;PCA 模块 2 PWM 寄存器。
PCA_PWM3 EQU    0F5H              ;PCA 模块 3 PWM 寄存器。
PCA_PWM4 EQU    0F6H              ;PCA 模块 4 PWM 寄存器。
PCA_PWM5 EQU    0F7H              ;PCA 模块 5 PWM 寄存器。

;PCA_PWMn:    7     6     5     4     3     2     1      0
;             -     -     -     -     -     -   EPCnH  EPCnL
;B7-B2: 保留
;B1(EPCnH): 在 PWM 模式下, 与 CCAPnH 组成 9 位数。
;B0(EPCnL): 在 PWM 模式下, 与 CCAPnL 组成 9 位数。
```

```
;------------------------
CCAPM0    EQU    0DAH                ;PCA 模块0的工作模式寄存器。
CCAPM1    EQU    0DBH                ;PCA 模块1的工作模式寄存器。
CCAPM2    EQU    0DCH                ;PCA 模块2的工作模式寄存器。
CCAPM3    EQU    0DDH                ;PCA 模块3的工作模式寄存器。
CCAPM4    EQU    0DEH                ;PCA 模块4的工作模式寄存器。
CCAPM5    EQU    0DFH                ;PCA 模块5的工作模式寄存器。

;CCAPMn:    7       6       5       4       3       2       1       0
;           -     ECOMn   CAPPn   CAPNn   MATn    TOGn    PWMn    ECCFn
;
;ECOMn = 1:允许比较功能。
;CAPPn = 1:允许上升沿触发捕捉功能。
;CAPNn = 1:允许下降沿触发捕捉功能。
;MATn  = 1:当匹配情况发生时，允许 CCON 中的 CCFn 置位。
;TOGn  = 1:当匹配情况发生时，CEXn 将翻转。
;PWMn  = 1:将 CEXn 设置为 PWM 输出。
;ECCFn = 1:允许 CCON 中的 CCFn 触发中断。

;ECOMn  CAPPn  CAPNn  MATn  TOGn  PWMn  ECCFn
;  0      0      0     0     0     0     0    00H 未启用任何功能。
;  x      1      0     0     0     0     x    21H 16位CEXn上升沿触发捕捉功能。
;  x      0      1     0     0     0     x    11H 16位CEXn下降沿触发捕捉功能。
;  x      1      1     0     0     0     x    31H 16位CEXn边沿(上、下沿)触发捕捉功能。
;  1      0      0     1     0     0     x    49H 16位软件定时器。
;  1      0      0     1     1     0     x    4DH 16位高速脉冲输出。
;  1      0      0     0     0     1     0    42H 8位 PWM。
;-----------------------------------------------------------------------------
;定义单片机管脚
LED_MCU_START       EQU P1.7
LED_PCA_INT0        EQU P1.6
LED_PCA_INT1        EQU P1.5
;-----------------------------------------------------------------------------
    ORG    0000H
    LJMP   MAIN
;-----------------------------------------------------------------------------
    ORG    0033H                   ;interrupt 6(0,1,2,3,4,5,6)
    LJMP   PCA_Interrupt
;-----------------------------------------------------------------------------
    ORG    0050H
MAIN:
    MOV    SP, #7FH
    CLR    LED_MCU_START           ;点亮 LED_MCU_START LED，表示程序正在运行
    LCALL PCA_Initiate             ;初始化 PCA
WAIT:
    SJMP   WAIT                    ;跳转到本行，无限循环。
```

```
;-------------------------------------------------------------------------
PCA_Initiate:
    MOV   CMOD, #10000000B ;PCA 在空闲模式下停止 PCA 计数器工作
                           ;PCA 时钟源为 fosc/12
                           ;禁止 PCA 计数器溢出(CH,CL 由 FFFFH 变为 0000H 时)中断
    MOV   CCON, #00H       ;CF = 0, 清 0 PCA 计数器溢出中断请求标志位
                           ;CR = 0, 不允许 PCA 计数器计数
                           ;清 0 PCA 各模块中断请求标志位, 如 CCF1, CCF0
    MOV   CL, #00H         ;清 0 PCA 计数器
    MOV   CH, #00H
;-------------------------
;设置模块 0
    MOV   CCAPM0, #11H          ;设置 PCA 模块 0 下降沿触发捕捉功能。
;   MOV   CCAPM0, #21H          ;如果送的是 #21h, 则 PCA 模块 0 为上升沿触发。
;-------------------------
;设置模块 1
    MOV   CCAPM1, #31H          ;设置 PCA 模块 1 上升沿 / 下降沿均可触发的捕捉功能。
;-------------------------
    SETB  EPCA_LVD              ;开 PCA 中断和 LVD(低压检测)中断共享的总中断控制位
    SETB  EA                    ;开整个单片机所有中断共享的总中断控制位
    SETB  CR                    ;启动 PCA 计数器(CH,CL)计数
    RET
;-------------------------------------------------------------------------
PCA_Interrupt:
    PUSH  ACC
    PUSH  PSW

    JNB   CCF0, Not_PCA0_Else_PCA1 ;如果 CCF0 不等于 1 就不是 PCA 模块 0 中断
                                   ;就直接去判是否是 PCA 模块 1 中断
    ;模块 0 中断服务程序
    CPL   LED_PCA_INT0             ;P1.6 LED 变化一次, 表示 PCA 模块 0 发生了一次中断
    CLR   CCF0                     ;清 PCA 模块 0 中断标志

Not_PCA0_Else_PCA1:
    JNB   CCF1, PCA_Interrupt_Exit ;如果 CCF1 不等于 1 就不是 PCA 模块 1 中断
                                   ;就立即退出
    ;模块 1 中断服务程序
    CPL   LED_PCA_INT1             ;P1.5 LED 变化一次, 表示 PCA 模块 1 发生了一次中断
    CLR   CCF1                     ;清 PCA 模块 1 中断标志

PCA_Interrupt_Exit:
    POP   PSW
    POP   ACC
    RETI
;-------------------------------------------------------------------------
    END
;-------------------------------------------------------------------------
```

PCA 功能做定时器的示例程序

```
;/* ---  STC International Limited ---------------------- */
;/* ---  宏晶科技     姚永平     2006/1/6     V1.0 ---------- */
;/* ---  PCA_5410_ASM_Timer ---------------------------- */
;/* ---  STC12C5412AD, STC12C5410AD, STC12C5408AD ------- */
;/* ---  STC12C5406AD, STC12C5404AD, STC12C5402AD ------- */
;/* ---  STC12C5052AD, STC12C4052AD, STC12C3052AD ------- */
;/* ---  STC12C2052AD, STC12C1052AD, STC12C0552AD ------- */
;/* ---  Mobile: 13922805190 --------------------------- */
;/* ---  Fax: 0755-82944243 ---------------------------- */
;/* ---  Tel: 0755-82948409 ---------------------------- */
;/* ---  Web: www.mcu-memory.com ------------------------ */
;-------------------------------------------------------------------------------
;使用 STC12C2052AD 系列单片机 PCA 功能做定时器的示例程序
;使用 STC12C5410AD 系列单片机 PCA 功能做定时器的示例程序
;晶振频率 Fosc = 18.432MHz，在P1.5 输出脉冲宽度为 1 秒钟的方波
;-------------------------------------------------------------------------------
;声明 STC12C2052AD和STC12C5410AD系列 MCU 特殊功能寄存器地址
IPH         EQU    0B7H             ;中断优先级高位寄存器
EPCA_LVD    EQU    IE.6             ;PCA 中断和 LVD(低压检测)中断共享的总中断控制位
CH          EQU    0F9H             ;PCA 计数器高8位。
CL          EQU    0E9H             ;PCA 计数器低8位。
;------------------------
CCON        EQU    0D8H             ;PCA 控制寄存器。
CCF0        EQU    CCON.0           ;PCA 模块0 中断标志，由硬件置位，必须由软件清0。
CCF1        EQU    CCON.1           ;PCA 模块1 中断标志，由硬件置位，必须由软件清0。
CCF2        EQU    CCON.2           ;PCA 模块2 中断标志，由硬件置位，必须由软件清0。
CCF3        EQU    CCON.3           ;PCA 模块3 中断标志，由硬件置位，必须由软件清0。
CCF4        EQU    CCON.4           ;PCA 模块4 中断标志，由硬件置位，必须由软件清0。
CCF5        EQU    CCON.5           ;PCA 模块5 中断标志，由硬件置位，必须由软件清0。
CR          EQU    CCON.6           ;1:允许 PCA 计数器计数，必须由软件清0。
CF          EQU    CCON.7           ;PCA计数器溢出(CH,CL由 FFFFH 变为 0000H)标志，
                                    ;PCA计数器溢出后由硬件置位,必须由软件清0。
;------------------------
CMOD        EQU    0D9H             ;PCA 工作模式寄存器。
;CMOD.7     CIDL: idle 状态时 PCA 计数器是否继续计数，0: 继续计数，1: 停止计数。

;CMOD.2     CPS1: PCA 计数器计数脉冲源选择位 1。
;CMOD.1     CPS0: PCA 计数器计数脉冲源选择位 0。
;           CPS1    CPS0
;            0       0      外部晶体频率/12。
;            0       1      外部晶体频率/2。
;            1       0      Timer 0 溢出脉冲，
;                           Timer 0 还可通过 AUXR寄存器设置成工作在 12T 或 1T 模式。
;            1       1      从 ECI/P3.4 脚输入的外部时钟。

;CMOD.0     ECF: PCA计数器溢出中断允许位，1-- 允许 CF(CCON.7) 产生中断。
```

```
;-----------------------
CCAP0H    EQU    0FAH              ;PCA 模块0的捕捉/比较寄存器高 8 位。
CCAP1H    EQU    0FBH              ;PCA 模块1的捕捉/比较寄存器高 8 位。
CCAP2H    EQU    0FCH              ;PCA 模块2的捕捉/比较寄存器高 8 位。
CCAP3H    EQU    0FDH              ;PCA 模块3的捕捉/比较寄存器高 8 位。
CCAP4H    EQU    0FEH              ;PCA 模块4的捕捉/比较寄存器高 8 位。
CCAP5H    EQU    0FFH              ;PCA 模块5的捕捉/比较寄存器高 8 位。

CCAP0L    EQU    0EAH              ;PCA 模块0的捕捉/比较寄存器低 8 位。
CCAP1L    EQU    0EBH              ;PCA 模块1的捕捉/比较寄存器低 8 位。
CCAP2L    EQU    0ECH              ;PCA 模块2的捕捉/比较寄存器低 8 位。
CCAP3L    EQU    0EDH              ;PCA 模块3的捕捉/比较寄存器低 8 位。
CCAP4L    EQU    0EEH              ;PCA 模块4的捕捉/比较寄存器低 8 位。
CCAP5L    EQU    0EFH              ;PCA 模块5的捕捉/比较寄存器低 8 位。

;-----------------------
PCA_PWM0 EQU    0F2H               ;PCA 模块0 PWM 寄存器。
PCA_PWM1 EQU    0F3H               ;PCA 模块1 PWM 寄存器。
PCA_PWM2 EQU    0F4H               ;PCA 模块2 PWM 寄存器。
PCA_PWM3 EQU    0F5H               ;PCA 模块3 PWM 寄存器。
PCA_PWM4 EQU    0F6H               ;PCA 模块4 PWM 寄存器。
PCA_PWM5 EQU    0F7H               ;PCA 模块5 PWM 寄存器。

;PCA_PWMn:    7      6      5      4      3      2      1      0
;             -      -      -      -      -      -    EPCnH  EPCnL
;B7-B2: 保留
;B1(EPCnH): 在 PWM 模式下，与 CCAPnH 组成 9 位数。
;B0(EPCnL): 在 PWM 模式下，与 CCAPnL 组成 9 位数。

;-----------------------
CCAPM0    EQU    0DAH              ;PCA 模块0的工作模式寄存器。
CCAPM1    EQU    0DBH              ;PCA 模块1的工作模式寄存器。
CCAPM2    EQU    0DCH              ;PCA 模块2的工作模式寄存器。
CCAPM3    EQU    0DDH              ;PCA 模块3的工作模式寄存器。
CCAPM4    EQU    0DEH              ;PCA 模块4的工作模式寄存器。
CCAPM5    EQU    0DFH              ;PCA 模块5的工作模式寄存器。

;CCAPMn:    7      6      5      4      3      2      1      0
;           -    ECOMn  CAPPn  CAPNn  MATn   TOGn   PWMn   ECCFn
;
;ECOMn = 1:允许比较功能。
;CAPPn = 1:允许上升沿触发捕捉功能。
;CAPNn = 1:允许下降沿触发捕捉功能。
;MATn  = 1:当匹配情况发生时，允许 CCON 中的 CCFn 置位。
;TOGn  = 1:当匹配情况发生时，CEXn 将翻转。
;PWMn  = 1:将 CEXn 设置为 PWM 输出。
;ECCFn = 1:允许 CCON 中的 CCFn 触发中断。
```

```
;ECOMn  CAPPn  CAPNn  MATn  TOGn  PWMn  ECCFn
;  0      0      0      0     0     0     0    00H 未启用任何功能。
;  x      1      0      0     0     0     x    21H 16 位 CEXn 上升沿触发捕捉功能。
;  x      0      1      0     0     0     x    11H 16 位 CEXn 下降沿触发捕捉功能。
;  x      1      1      0     0     0     x    31H 16 位 CEXn 边沿(上、下沿)触发捕捉功能。
;  1      0      0      1     0     0     x    49H 16 位软件定时器。
;  1      0      0      1     1     0     x    4DH 16 位高速脉冲输出。
;  1      0      0      0     0     1     0    42H 8 位 PWM。
;--------------------------------------------------------------------------
;定义单片机管脚
LED_MCU_START       EQU P1.7
LED_5mS_Flashing    EQU P1.6
LED_1S_Flashing     EQU P1.5
;--------------------------------------------------------------------------
;定义常量
;Channe0_5mS_H, Channe0_5mS_L 的计算方法见 PCA 中断服务程序内的注释
Channe0_5mS_H       EQU 1EH      ;模块 0 5mS 定时常数高位
Channe0_5mS_L       EQU 00H      ;模块 0 5mS 定时常数低位
;--------------------------------------------------------------------------
;定义变量
Counter              EQU 30H      ;声明一个计数器，用来计数中断的次数
;--------------------------------------------------------------------------
    ORG    0000H
    LJMP   MAIN
;--------------------------------------------------------------------------
    ORG    0033H                  ;interrupt 6(0,1,2,3,4,5,6)
    LJMP   PCA_interrupt
;--------------------------------------------------------------------------
    ORG    0050H
MAIN:
    CLR    LED_MCU_START          ;点亮 MCU 开始工作指示灯
    MOV    SP, #7FH
    MOV    Counter, #0            ;清 Counter 计数器

    ACALL PCA_Initiate            ;初始化 PCA
WAIT:
    SJMP   WAIT                   ;跳转到本行，无限循环。
;--------------------------------------------------------------------------
PCA_Initiate:
    MOV    CMOD, #10000000B ;PCA 在空闲模式下停止 PCA 计数器工作
                            ;PCA 时钟源为 fosc/12
                            ;禁止 PCA 计数器溢出(CH,CL 由 FFFFH 变为 0000H 时)中断
    MOV    CCON, #00H       ;CF = 0，清 0 PCA 计数器溢出中断请求标志位
                            ;CR = 0，不允许 PCA 计数器计数
                            ;清 0 PCA 各模块中断请求标志位，如 CCF1，CCF0
    MOV    CL, #00H         ;清 0 PCA 计数器
    MOV    CH, #00H
```

```
;-------------------------
    ;Channe0_5mS_H, Channe0_5mS_L 的计算方法见 PCA 中断服务程序内的注释
    MOV   CCAP0L, #Channe0_5mS_L ;给 PCA 模块0 的 CCAP0L 置初值
    MOV   CCAP0H, #Channe0_5mS_H ;给 PCA 模块0 的 CCAP0H 置初值
    MOV   CCAPM0, #49H     ;设置 PCA 模块0 为16位软件定时器,ECCF0=1允许PCA模块0中断
     ;当[CH, CL]==[CCAP0H, CCAP0L]时，产生中断请求，CCF0=1，请求中断
    SETB  EPCA_LVD                ;开 PCA 中断和 LVD(低压检测)中断共享的总中断控制位
    SETB  EA                      ;开整个单片机所有中断共享的总中断控制位
    SETB  CR                      ;启动 PCA 计数器(CH,CL)计数
    RET
;-----------------------------------------------------------------------------
PCA_Interrupt:
    PUSH  ACC
    PUSH  PSW

    CPL   LED_5mS_Flashing   ;本程序PCA模块0每5mS中断一次，每次进中断将该灯状态取反

    ;在本程序中[CH,CL]每12个时钟脉冲加1，当[CH,CL] 增加到等于 [CCAP0H, CCAP0L]时
    ;CCF0=1，产生中断请求。如果每次 PCA 模块0 中断后，在中断服务程序中给
    ;[CCAP0H, CCAP0L] 增加一个相同的数值，那么下一次中断来临的间隔时间 T 也是相
    ;同的。本程序中这个"相同的数值" 就是 Channe0_5mS_H, Channe0_5mS_L

    ;举例：时钟频率 Fosc = 18.432MHz, PCA 计数器计数 1E00H 次才是 5mS。
    ;计算PCA 计数器计数多少次：
    ;     Channe0_5mS_H, Channe0_5mS_L = T/( (1/Fosc)*12 )
    ;                                  = 0.005/ ( (1/18432000)*12 )
    ;                                  = 7680 (10进制数)
    ;                                  = 1E00H (16进制数)
    ;     即 Channe0_5mS_H = 1EH, Channe0_5mS_L = 00H
    ;
    ;    Channe0_5mS_H, Channe0_5mS_L：每次给 [CCAP0H,CCAP0L] 增加的数值(步长)

    MOV   A, #Channe0_5mS_L      ;给[CCAP0H, CCAP0L] 增加一个数值
    ADD   A, CCAP0L
    MOV   CCAP0L, A
    MOV   A, #Channe0_5mS_H
 ADDC  A, CCAP0H
    MOV   CCAP0H, A
    CLR   CCF0                    ;清 PCA 模块0 中断标志

    INC   Counter                 ;中断次数计数器 + 1
    MOV   A, Counter
    CLR   C
    SUBB  A, #200                 ;检测是否中断了 200 次（1秒）
    JC    PCA_Interrupt_Exit      ;有借位，表示 Counter 小于 200，立即跳转退出
```

```
    MOV    Counter, #0           ;已中断了 200 次，清 0 中断次数计数器
    CPL    LED_1S_Flashing       ;在 LED_1S_Flashing 输出脉冲宽度为 1 秒钟的方波

PCA_Interrupt_Exit:
    POP    PSW
    POP    ACC
    RETI
;-----------------------------------------------------------------------------
    END
```

STC12C5410AD 系列单片机 PWM 输出 C 语言示例

PWM 输出 C 语言示例

```
#include<reg52.h>
sfr  CCON   = 0xD8;
sfr  CMOD   = 0xD9;
sfr  CL     = 0xE9;
sfr  CH     = 0xF9;
sfr  CCAP0L = 0xEA;
sfr  CCAP0H = 0xFA;
sfr  CCAPM0 = 0xDA;
sfr  CCAPM1 = 0xDB;
sbit  CR = 0xDE;
void main(void)
{
    CMOD = 0x02; // Setup PCA timer
    CL = 0x00;
    CH = 0x00;
    CCAP0L = 0xc0;  //Set the initial value same as CCAP0H
    CCAP0H = 0xc0;  //25% Duty Cycle
    CCAPM0 = 0x42;  //0100,0010  Setup PCA module 0 in PWM mode
    CR = 1; //Start PCA Timer.
    while(1){};
 }
```

STC12C5410AD 系列单片机 PCA/PWM 新增特殊功能寄存器声明

```
;声明 STC12C2052AD 和 STC12C5410AD 系列 MCU 特殊功能寄存器地址
IPH        EQU    0B7H                ;中断优先级高位寄存器
EPCA_LVD   EQU    IE.6                ;PCA 中断和 LVD(低压检测)中断共享的总中断控制位
CH         EQU    0F9H                ;PCA 计数器高 8 位。
CL         EQU    0E9H                ;PCA 计数器低 8 位。
;-------------------------
CCON       EQU    0D8H                ;PCA 控制寄存器。
CCF0       EQU    CCON.0              ;PCA 模块 0 中断标志，由硬件置位，必须由软件清 0。
CCF1       EQU    CCON.1              ;PCA 模块 1 中断标志，由硬件置位，必须由软件清 0。
CCF2       EQU    CCON.2              ;PCA 模块 2 中断标志，由硬件置位，必须由软件清 0。
CCF3       EQU    CCON.3              ;PCA 模块 3 中断标志，由硬件置位，必须由软件清 0。
CCF4       EQU    CCON.4              ;PCA 模块 4 中断标志，由硬件置位，必须由软件清 0。
CCF5       EQU    CCON.5              ;PCA 模块 5 中断标志，由硬件置位，必须由软件清 0。
CR         EQU    CCON.6              ;1:允许 PCA 计数器计数，必须由软件清 0。
CF         EQU    CCON.7              ;PCA 计数器溢出(CH,CL 由 FFFFH 变为 0000H)标志，
                                      ;PCA 计数器溢出后由硬件置位,必须由软件清 0。
;-------------------------
CMOD       EQU    0D9H                ;PCA 工作模式寄存器。
;CMOD.7       CIDL: idle 状态时 PCA 计数器是否继续计数，0: 继续计数，1: 停止计数。

;CMOD.2       CPS1: PCA 计数器计数脉冲源选择位 1。
;CMOD.1       CPS0: PCA 计数器计数脉冲源选择位 0。
;             CPS1    CPS0
;              0        0     外部晶体频率 /12。
;              0        1     外部晶体频率 /2。
;              1        0     Timer 0 溢出脉冲，
;                             Timer 0 还可通过 AUXR 寄存器设置成工作在 12T 或 1T 模式。
;              1        1     从 ECI/P3.4 脚输入的外部时钟。

;CMOD.0       ECF: PCA 计数器溢出中断允许位，1-- 允许 CF(CCON.7) 产生中断。
;-------------------------
CCAP0H     EQU    0FAH                ;PCA 模块 0 的捕捉 / 比较寄存器高 8 位。
CCAP1H     EQU    0FBH                ;PCA 模块 1 的捕捉 / 比较寄存器高 8 位。
CCAP2H     EQU    0FCH                ;PCA 模块 2 的捕捉 / 比较寄存器高 8 位。
CCAP3H     EQU    0FDH                ;PCA 模块 3 的捕捉 / 比较寄存器高 8 位。
CCAP4H     EQU    0FEH                ;PCA 模块 4 的捕捉 / 比较寄存器高 8 位。
CCAP5H     EQU    0FFH                ;PCA 模块 5 的捕捉 / 比较寄存器高 8 位。

CCAP0L     EQU    0EAH                ;PCA 模块 0 的捕捉 / 比较寄存器低 8 位。
CCAP1L     EQU    0EBH                ;PCA 模块 1 的捕捉 / 比较寄存器低 8 位。
CCAP2L     EQU    0ECH                ;PCA 模块 2 的捕捉 / 比较寄存器低 8 位。
CCAP3L     EQU    0EDH                ;PCA 模块 3 的捕捉 / 比较寄存器低 8 位。
CCAP4L     EQU    0EEH                ;PCA 模块 4 的捕捉 / 比较寄存器低 8 位。
CCAP5L     EQU    0EFH                ;PCA 模块 5 的捕捉 / 比较寄存器低 8 位。
```

```
;-----------------------
PCA_PWM0 EQU   0F2H             ;PCA 模块0 PWM 寄存器。
PCA_PWM1 EQU   0F3H             ;PCA 模块1 PWM 寄存器。
PCA_PWM2 EQU   0F4H             ;PCA 模块2 PWM 寄存器。
PCA_PWM3 EQU   0F5H             ;PCA 模块3 PWM 寄存器。
PCA_PWM4 EQU   0F6H             ;PCA 模块4 PWM 寄存器。
PCA_PWM5 EQU   0F7H             ;PCA 模块5 PWM 寄存器。

;PCA_PWMn:    7      6      5      4      3      2      1      0
;             -      -      -      -      -      -    EPCnH  EPCnL
;B7-B2: 保留
;B1(EPCnH): 在 PWM 模式下，与 CCAPnH 组成 9 位数。
;B0(EPCnL): 在 PWM 模式下，与 CCAPnL 组成 9 位数。

;-----------------------
CCAPM0   EQU   0DAH             ;PCA 模块0的工作模式寄存器。
CCAPM1   EQU   0DBH             ;PCA 模块1的工作模式寄存器。
CCAPM2   EQU   0DCH             ;PCA 模块2的工作模式寄存器。
CCAPM3   EQU   0DDH             ;PCA 模块3的工作模式寄存器。
CCAPM4   EQU   0DEH             ;PCA 模块4的工作模式寄存器。
CCAPM5   EQU   0DFH             ;PCA 模块5的工作模式寄存器。

;CCAPMn:    7      6      5      4      3      2      1      0
;           -    ECOMn  CAPPn  CAPNn  MATn   TOGn   PWMn   ECCFn
;
;ECOMn = 1:允许比较功能。
;CAPPn = 1:允许上升沿触发捕捉功能。
;CAPNn = 1:允许下降沿触发捕捉功能。
;MATn  = 1:当匹配情况发生时，允许 CCON 中的 CCFn 置位。
;TOGn  = 1:当匹配情况发生时，CEXn 将翻转。
;PWMn  = 1:将 CEXn 设置为 PWM 输出。
;ECCFn = 1:允许 CCON 中的 CCFn 触发中断。
;ECOMn  CAPPn  CAPNn  MATn  TOGn  PWMn  ECCFn
;  0      0      0      0     0     0     0    00H 未启用任何功能。
;  x      1      0      0     0     0     x    21H 16位CEXn上升沿触发捕捉功能。
;  x      0      1      0     0     0     x    11H 16位CEXn下降沿触发捕捉功能。
;  x      1      1      0     0     0     x    31H 16位CEXn边沿(上、下沿)触发捕捉功能。
;  1      0      0      1     0     0     x    49H 16位软件定时器。
;  1      0      0      1     1     0     x    4DH 16位高速脉冲输出。
;  1      0      0      0     0     1     0    42H 8位 PWM。
;-------------------------------------------------------------------------------
```

STC12C5410AD 系列单片机 PCA 功能 PWM 输出程序

```
;STC12C5410AD 系列单片机 PCA 功能 PWM 示例程序，使用18.432MHz 晶振。
;-----------------------------------------------------------------
#include <..\PCA_5410AD_SFR.ASM>      ;定义 PCA 特殊功能寄存器
;-----------------------------------------------------------------

;定义常量
;pulse_width_MAX = pulse_width_MIN 时，输出脉冲宽度不变。
pulse_width_MAX    EQU    0F0H         ;PWM 脉宽最大值，占空比 = 93.75%
pulse_width_MIN    EQU    10H          ;PWM 脉宽最小值，占空比 = 6.25%
step               EQU    38H          ;PWM 脉宽变化步长
;-----------------------------------------------------------------
;定义变量
pulse_width      EQU   30H
;-----------------------------------------------------------------
    ORG 0000H
    AJMP   main
;-----------------------------------------------------------------
    ORG 0050H
main:
    MOV    SP, #0E0H

    ACALL PCA_init
main_loop:
    ACALL PWM
    SJMP   main_loop

;-----------------------------------------------------------------
PCA_init:
    MOV    CMOD, #80H;             ;PCA 在空闲模式下停止 PCA 计数器工作
                                   ;PCA 时钟模式为 fosc/12
                                   ;禁止 PCA 计数器溢出中断
    MOV    CCON, #00H              ;禁止 PCA 计数器工作，清除中断标志、计数器溢出标志
    MOV    CL, #00H                ;清0计数器
    MOV    CH, #00H
;------------------------
;设置模块0为 8 位 PWM 输出模式，PWM 无需中断支持。脉冲在P3.7(第11脚)输出
    MOV    CCAPM0, #42H            ;*** 示例程序核心语句，---->0100,0010
    MOV    PCA_PWM0, #00H          ;*** 示例程序核心语句
;   MOV    PCA_PWM0, #03H          ;释放本行注释，PWM 输出就一直是 0，无脉冲。

;------------------------
;设置模块1为 8 位 PWM 输出模式，PWM 无需中断支持。脉冲在P3.5(第9脚)输出
    MOV    CCAPM1, #42H            ;*** 示例程序核心语句，---->0100,0010
    MOV    PCA_PWM1, #00H          ;*** 示例程序核心语句
```

```
;   MOV   PCA_PWM1, #03H         ;释放本行注释，PWM 输出就一直是 0，无脉冲。

    SETB  EPCA_LVD               ;开 PCA 中断
    SETB  EA                     ;开总中断
    SETB  CR                     ;将 PCA 计数器打开
    RET

;--------------------------------------------------------------

PWM:                              ;用示波器进行观察较为理想。

     ;逐渐变亮。
    MOV   A, #pulse_width_MIN     ;为输出脉冲宽度设置初值。
    MOV   pulse_width, A          ;pulse_width 数字越大脉宽越窄，P3.5 的 LED 越亮。
PWM_loop1:
    MOV   A, pulse_width          ;判是否到达最大值。
    CLR   C
    SUBB  A, #pulse_width_MAX
    JNC   PWM_a                   ;到达最大值就转到逐渐变暗。
     MOV  A, pulse_width          ;设置脉冲宽度。数字越大、脉宽越窄、LED 越亮。
    MOV   CCAP0H, A               ;*** 示例程序核心语句
    MOV   CCAP1H, A               ;*** 示例程序核心语句

    CPL   A                       ;用 P1 口的 LED 显示占空比，
    MOV   P1, A                   ;占空比 = ( pulse_width/256 ) * 100%  。

    MOV   A, pulse_width          ;计算下一次输出脉冲宽度数值。
    ADD   A, #step
    MOV   pulse_width, A
    ACALL delay                   ;在一段时间内保持输出脉冲宽度不变。
    SJMP  PWM_loop1

PWM_a:
     ;逐渐变暗。
    MOV   A, #pulse_width_MAX     ;为输出脉冲宽度设置初值。
    MOV   pulse_width, A          ;pulse_width 数字越大脉宽越窄，P3.5 的 LED 越亮。
PWM_loop2:
    MOV   A, pulse_width          ;判是否到达最小值。
    CLR   C
    SUBB  A, #pulse_width_MIN
    JC    PWM_b                   ;到达最小值就返回。
    JZ    PWM_b                   ;到达最小值就返回。
     MOV  A, pulse_width          ;设置脉冲宽度。数字越大、脉宽越窄、LED 越亮。
    MOV   CCAP0H, A               ;*** 示例程序核心语句
    MOV   CCAP1H, A               ;*** 示例程序核心语句

    CPL   A                       ;用 P1 口的 LED 显示占空比，
```

```
    MOV    P1, A                ;占空比 = ( pulse_width/256 ) * 100%  。

    MOV    A, pulse_width       ;计算下一次输出脉冲宽度数值。
    CLR    C
    SUBB   A, #step
    MOV    pulse_width, A
    ACALL delay                 ;在一段时间内保持输出脉冲宽度不变。
    SJMP   PWM_loop2
PWM_b:
    RET
;-------------------------------------------------------------

delay:
    CLR    A
    MOV    R1, A
    MOV    R2, A
    MOV    R3, #80H
delay_loop:
    NOP
    NOP
    NOP
    DJNZ   R1, delay_loop
    DJNZ   R2, delay_loop
    DJNZ   R3, delay_loop
    RET
;-------------------------------------------------------------
    END
```

STC12C5410AD 系列单片机 PCA 的高速脉冲输出

```
;*****************************************************************************
;                  输出 125.0KHz 的脉冲(晶体频率 = 33.000MHz)
;
;示例程序: 使用 功能，在 P3.5(第 9 脚)输出
;           125.0KHz 的方脉冲。
;-----------------------------------------------------------------------------
;     程序中定义的常量 CCAPnL_Value 决定了 PCA 模块 n 输出脉冲的频率 f：
;         f =  Fosc / (4 * CCAPnL_Value )
;            式中 Fosc = 晶体频率
;         CCAPnL_Value  =  Fosc / (4 * f)
;
;     如算出的结果不是整数，则进行取整 CCAPnL_Value =  INT(Fosc / (4 * f) + 0.5)
;         INT() 为取整数运算，直接去掉小数。
;*****************************************************************************
;定义 STC12C5410 系列 MCU 特殊功能寄存器
IPH        EQU    0B7H                ;中断优先级高位寄存器

EPCA_LVD EQU    IE.6                ;PCA/LVD 中断允许位。
                  ;要打开 PCA 中断还要打开相应的 ECF，ECCF0，ECCF1 位
                  ;要打开 LVD 中断还要打开相应的 ELVDI 位
CH         EQU    0xF9                ;PCA 计数器高 8 位。
CL         EQU    0xE9                ;PCA 计数器低 8 位。

;------------------------
CCON       EQU    0D8H                ;PCA 控制寄存器。
CCF0       EQU    CCON.0              ;PCA 模块 0 中断标志，由硬件置位，必须由软件清 0。
CCF1       EQU    CCON.1              ;PCA 模块 1 中断标志，由硬件置位，必须由软件清 0。
CR         EQU    CCON.6              ;1:允许 PCA 计数器计数，必须由软件清 0。
CF         EQU    CCON.7              ;PCA 计数器溢出标志,由硬件或软件置位,必须由软件清 0。

;------------------------
CMOD       EQU    0D9H                ;PCA 工作模式寄存器。
;CMOD.7       CIDL: idle 状态时 PCA 计数器是否继续计数，0: 继续计数，1: 停止计数。

;CMOD.2       CPS1: PCA 计数器脉冲源选择位 1。
;CMOD.1       CPS0: PCA 计数器脉冲源选择位 0。
;             CPS1    CPS0
;              0       0      内部时钟，fosc/12。
;              0       1      内部时钟，fosc/2。
;              1       0      Timer0 溢出。
;              1       1      由 ECI/P3.4 脚输入的外部时钟。

;CMOD.0       ECF: PCA 计数器溢出中断允许位，1--允许 CF(CCON.7) 产生中断。
;------------------------
```

```
CCAP0H    EQU    0FAH              ;PCA 模块 0 的捕捉 / 比较寄存器高 8 位。
CCAP1H    EQU    0FBH              ;PCA 模块 1 的捕捉 / 比较寄存器高 8 位。
CCAP0L    EQU    0EAH              ;PCA 模块 0 的捕捉 / 比较寄存器低 8 位。
CCAP1L    EQU    0EBH              ;PCA 模块 1 的捕捉 / 比较寄存器低 8 位。

;-----------------------
PCA_PWM0 EQU    0F2H               ;PCA 模块 0 PWM 寄存器。
PCA_PWM1 EQU    0F3H               ;PCA 模块 1 PWM 寄存器。

;PCA_PWMn:    7      6      5      4      3      2      1      0
;             -      -      -      -      -      -    EPCnH  EPCnL
;B7-B2: 保留
;B1(EPCnH): 在 PWM 模式下，与 CCAPnH 组成 9 位数。
;B0(EPCnL): 在 PWM 模式下，与 CCAPnL 组成 9 位数。

;-----------------------
CCAPM0    EQU    0DAH              ;PCA 模块 0 的工作模式寄存器。
CCAPM1    EQU    0DBH              ;PCA 模块 1 的工作模式寄存器。

;CCAPMn:    7      6      5      4      3      2      1      0
;           -    ECOMn  CAPPn  CAPNn  MATn   TOGn   PWMn   ECCFn
;
;ECOMn  = 1:允许比较功能。
;CAPPn  = 1:允许上升沿触发捕捉功能。
;CAPNn  = 1:允许下降沿触发捕捉功能。
;MATn   = 1:当匹配情况发生时，允许 CCON 中的 CCFn 置位。
;TOGn   = 1:当匹配情况发生时，CEXn 将翻转。
;PWMn   = 1:将 CEXn 设置为 PWM 输出。
;ECCFn = 1:允许 CCON 中的 CCFn 触发中断。

;ECOMn CAPPn CAPNn MATn TOGn PWMn ECCFn
;  0     0     0     0    0    0    0    0x00   未启用任何功能。
;  x     1     0     0    0    0    x    0x21   16 位 CEXn 上升沿触发捕捉功能。
;  x     0     1     0    0    0    x    0x11   16 位 CEXn 下降沿触发捕捉功能。
;  x     1     1     0    0    0    x    0x31   16 位 CEXn 边沿(上、下沿)触发捕捉功能。
;  1     0     0     1    0    0    x    0x49   16 位软件定时器。
;  1     0     0     1    1    0    x    0x4d   16 位高速脉冲输出。
;  1     0     0     0    0    1    0    0x42   8 位 PWM。

;-----------------------------------------------------------
;定义常量 CCAPnL_Value
;CCAPnL_Value 决定了模块 1 输出脉冲的频率 f：
;         f =  Fosc / (4 * CCAPnL_Value )
;    式中 Fosc = 晶体频率
;    或  CCAPnL_Value  =  INT(Fosc / (4 * f) + 0.5)
;         INT() 为取整数运算。
;
```

```
;    假定 fosc = 20MHz时，要求 PCA 高速脉冲输出 125KHz 的方波:
;        CCAPnL_Value = INT( 20000000/4/125000 + 0.5)
;                     = INT( 40 + 0.5)
;                     = INT( 40.5 )
;                     = 40
;                     = 28H
;    输出脉冲的频率 f  = 20000000/4/40
;                     = 125000 (125.0KHz)

;CCAPnL_Value   EQU 25H      ;25H = 37, fosc = 18.432MHz时, 高速脉冲输出 = 124.540KHz
;CCAPnL_Value   EQU 28H      ;28H = 40, fosc = 20MHz时, 高速脉冲输出 = 125KHz
CCAPnL_Value    EQU 42H      ;42H = 66, fosc = 33MHz时, 高速脉冲输出 = 125KHz
;-----------------------------------------------------------
    ORG 0000H
    AJMP  main
;-----------------------------------------------------------
    ORG  0033H                  ;interrupt 6
PCA_interrupt:
    PUSH  ACC                   ;4 Clock
    PUSH  PSW                   ;4 Clock

    CLR   CCF1                  ;1 Clock, 清 PCA 模块 1 中断标志

    MOV   A, #CCAPnL_Value ;2 Clock
    ADD   A, CCAP1L             ;3 Clock
    MOV   CCAP1L, A             ;3 Clock
    CLR   A                     ;1 Clock
    ADDC  A, CCAP1H             ;3 Clock
    MOV   CCAP1H, A             ;3 Clock

    POP   PSW                   ;3 Clock
    POP   ACC                   ;3 Clock
    RETI                        ;4 Clock

;此中断服务程序共用 34 Clock, 进入中断服务程序还要数个 Clock
;-----------------------------------------------------------
    ORG 0060H
main:
    MOV   SP, #0E0H             ;设置堆栈指针
    ACALL PCA_init              ;调用 PCA 初始化程序

main_loop:
    NOP
    NOP
    NOP
    SJMP  main_loop
;-----------------------------------------------------------
```

```
PCA_init:                          ;PCA 初始化程序
    MOV    CMOD, #00000010B        ;02H, PCA 计数器在空闲模式下继续工作, CIDL = 0
                        ;PCA 计数器计数脉冲来源为系统时钟源 fosc/2, CPS1, CPS0 = (0,1)
                        ;禁止 PCA 计数器(CH, CL)计数溢出(CH, CL=0000H)中断, ECF = 0
    MOV    CCON, #00H              ;清除 PCA 计数器(CH, CL)计数溢出中断标志, CF = 0
                                   ;停止 PCA 计数器(CH, CL)计数, CR = 0
                                   ;清除 模块1 中断标志, CCF1 = 0
                                   ;清除 模块0 中断标志, CCF0 = 0
    MOV    CH, #00H                ;清0 PCA 计数器高 8 位
    MOV    CL, #00H                ;清0 PCA 计数器低 8 位
;------------------------
;设置模块1为高速脉冲输出模式, 脉冲在P3.5(第9脚)输出
    MOV    CCAPM1, #01001101B      ;4DH, 设置 PCA 模块1为高速脉冲输出模式,允许触发中断
;CCAPMn:     7       6       5       4       3       2       1       0
;            -     ECOMn   CAPPn   CAPNn   MATn    TOGn    PWMn    ECCFn
;            0       1       0       0       1       1       0       1

    MOV    CCAP1L, #CCAPnL_Value ;给模块1置初值, 此句不可少
    MOV    CCAP1H, #0 ;给模块1置初值, 此句不可少

    ;其它中断服务可能会使模块1 高速脉冲输出的某个周期突然变得很大,因此必须将
    ;PCA 中断的优先级设置为唯一的最高级, 其它中断的优先级都要比它低。
    MOV    IPH, #01000000B         ;PCA 中断的优先级设置为唯一的最高级
    MOV    IP, #01000000B

    SETB   EPCA_LVD                ;开 PCA 中断
    SETB   EA                      ;开总中断
    SETB   CR                      ;将 PCA 计数器打开
    RET

;-----------------------------------------------------------
    END
;-----------------------------------------------------------
```

STC12C5410AD 系列单片机应用注意事项

关于复位电路：

晶振频率在 20M 以下时:可以不用外部复位电路，原复位电路可以保留，也可以不用，不用时复位脚可直接短到地。不过建议设计时 PCB 板上保留 R/C 复位电路，实际使用时再决定用或不用。

关于时钟：

如果使用内部 R/C 振荡器时钟(4MHz～8MHz,制造误差加温漂),XTAL1 和 XTAL2 脚浮空.

如果外部时钟频率在 33MHz 以上时,建议直接使用外部有源石英晶体振荡器,时钟从 XTAL1 脚输入,XTAL2 脚必须浮空.

老版本应用注意事项(现新版本 C 版本已经改好，并开始大量供货)：

★空闲模式（IDLE）不要用，新版本 C 版本已修改好

★对普通可位寻址的特殊功能寄存器的位(位地址: 00H - 7FH),用 JBC RAM_Bit 无任何问题

★对可位寻址的特殊功能寄存器的位(80H - FFH)

不要用 JBC SFR_Bit 指令

用 JB SFR_Bit / CLR SFR_Bit 指令取代

举例:

定时器 0、定时器 1 使用时建议直接用中断方式，与普通 8051 单片机完全兼容

但如不用中断，而用查询方式，建议用

JB TF0 查询 / 再 CLR TF0；JB TF1 查询 / 再 CLR TF1。

不要用

JBC TF0 / JBC TF1

此问题新版本正在修正中，但实际并不影响用户使用。

原因：

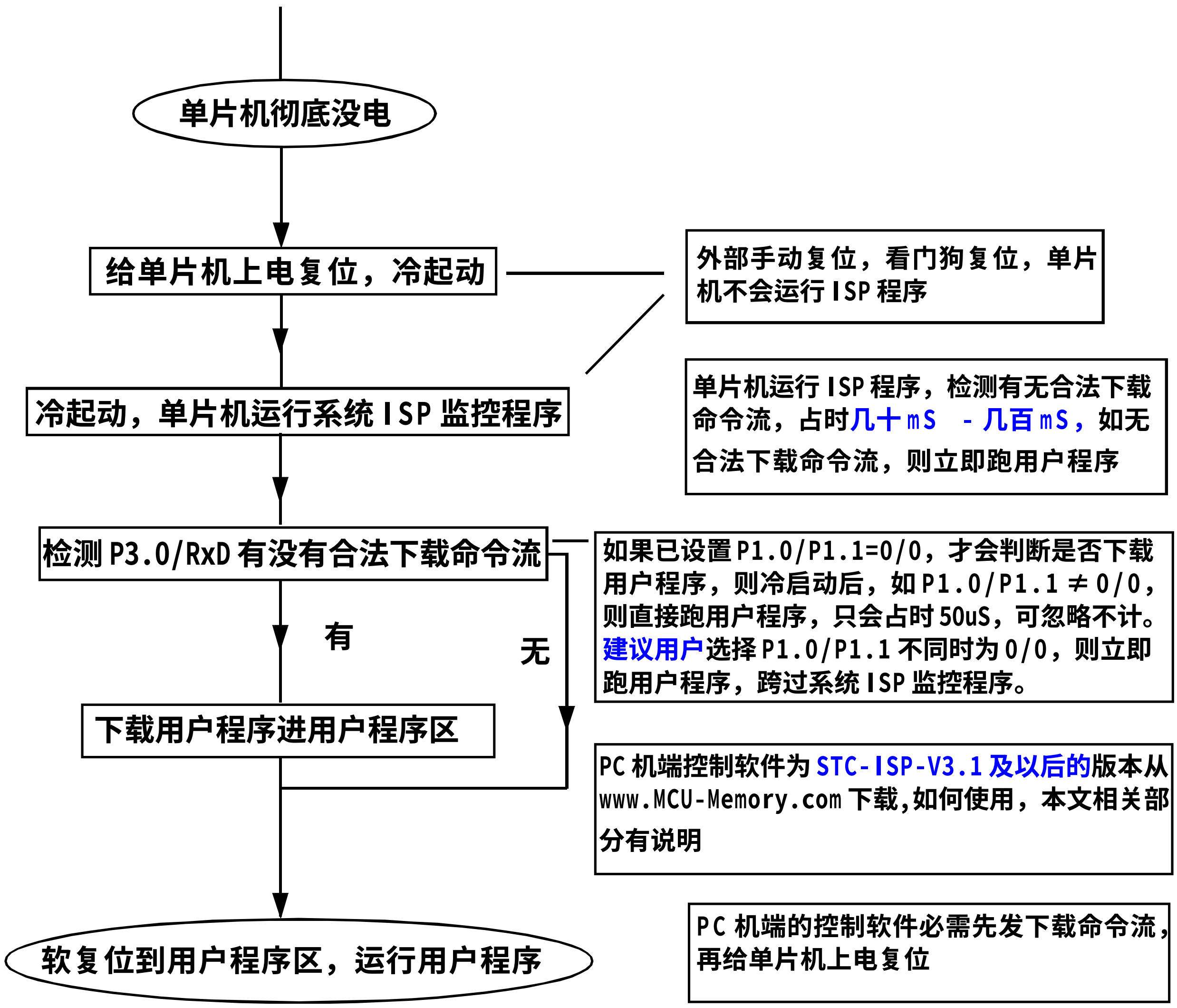
主要是特殊功能寄存器的整个 8 位值被读出后,如在我们判断该位为 1 后,将该位清零,并将整个 8 位值再一起写回之前,它的部分其它位如已由硬件设置发生变化,而我们 JBC 判断后再写回时,新的值被破坏,被写成了老的值。

如果该位是 0,则不会再写回,就不会发生此事。

如果该位变成了 1,该特殊功能寄存器的其它位在此瞬间未发生变化,再一起写回，也无问题。

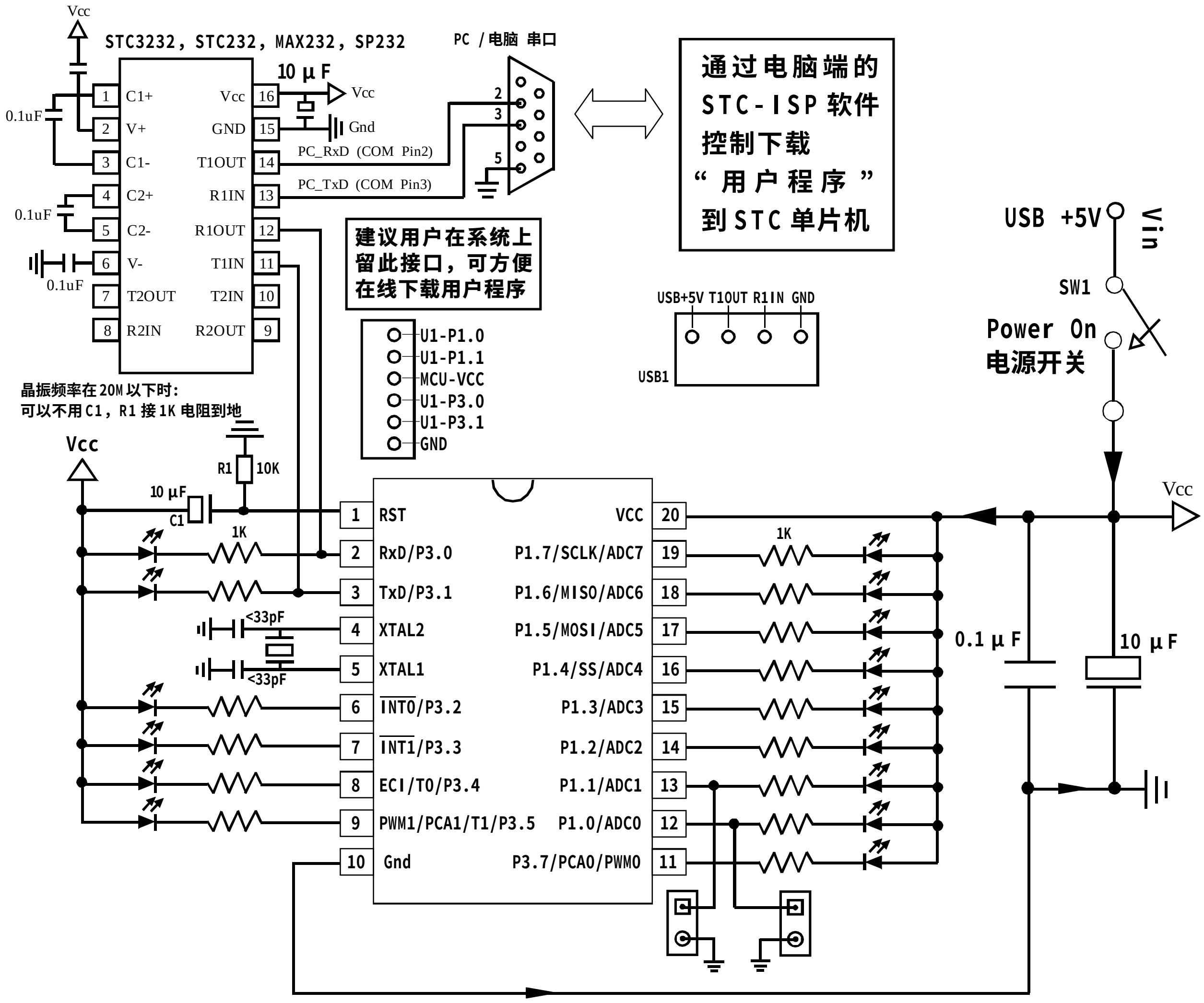
如果该位变成了 1,该特殊功能寄存器的其它位在此瞬间发生变化,其它在此瞬间发生变化的位就会被破坏,又被写成老的值。

STC12C5410AD 系列单片机 ISP 编程原理 使用说明



STC 12C5410AD 系列单片机在系统可编程的使用

--- 将用户代码下载进单片机内部，不用编程器



STC12C5410AD 系列单片机具有在系统可编程（ISP）特性，ISP 的好处是：省去购买通用编程器，单片机在用户系统上即可下载／烧录用户程序，而无须将单片机从已生产好的产品上拆下，再用通用编程器将程序代码烧录进单片机内部。有些程序尚未定型的产品可以一边生产，一边完善，加快了产品进入市场的速度，减小了新产品由于软件缺陷带来的风险。由于可以在用户的目标系统上将程序直接下载进单片机看运行结果对错，故无须仿真器。

STC12 系列单片机内部固化有 ISP 系统引导固件，配合 PC 端的控制程序即可将用户的程序代码下载进单片机内部，故无须编程器(速度比通用编程器快，几秒一片)。

如何获得及使用 STC 提供的 ISP 下载工具（STC-ISP.exe 软件）：

(1). 获得 STC 提供的 ISP 下载工具（软件）

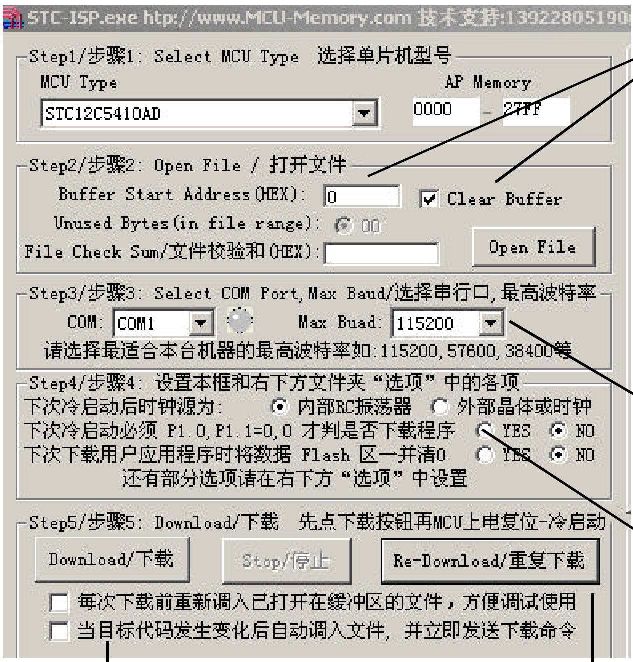
登陆 www.MCU-Memory.com 网站，从 STC 半导体专栏下载 PC（电脑）端的 ISP 程序，然后将其自解压，再安装即可（执行 setup.exe），注意随时更新软件。

(2). 使用 STC-ISP 下载工具（软件），请随时更新，目前已到 Ver3.8 以上版本，支持 *.bin, *.hex(Intel 16 进制格式)文件。

请随时注意升级 PC（电脑）端的 ISP 程序。

(3).STC12C5410AD 系列单片机出厂时就已完全加密。需要单片机内部的电放光后上电复位(冷起动)才运行系统 ISP 程序，如从 P3.0/RxD 检测到合法的下载命令流就下载用户程序，如检测不到就系统复位到用户程序区。

(4).如果用户板上 P3.0/RxD，P3.1/Txd 接了 RS-485 等电路，下载时需要将其断开。用户系统接了 RS-485 等通信电路，推荐在选项中选择下次冷启动时需 P1.0/P1.1=0/0 才判是否下载程序。



第一次调文件进缓冲区，要清缓冲区。
要调几个文件进缓冲区，如 EEPROM 里的数据文件需要和应用程序文件一次同时 ISP 下载编程进单片机：

除每次均要指定缓冲区起始地址外，第二次及以后不能清缓冲区
如可将要写入 EEPROM 区的数据文件调入从缓冲区 2800H 开始的地方，并不清缓冲区，然后和应用程序一起写入

用户根据实际使用效果选择限制最高通信波特率，如 57600，38400，19200

如 P3.0/P3.1 外接 RS-485/RS-232 等通信电路，建议选择如 P1.0/P1.1 不同时等于 0/0，则直接运行用户程序，跨过系统 ISP 引导程序
新的设置冷启动（彻底停电后再上电）后才生效

开发调试时，可考虑选择此项

大批量生产时使用

Step1/ 步骤 1：选择你所使用的单片机型号，如 STC12C5410，STC12C5410AD 等
Step2/ 步骤 2：打开文件，要烧录用户程序，必须调入用户的程序代码（*.bin，*.hex）
Step3/ 步骤 3：选择串行口，你所使用的电脑串口，如串行口 1--COM1， 串行口 2--COM2，...
有些新式笔记本电脑没有 RS-232 串行口，可买一条 USB-RS232 转接器，人民币 50 元左右。
有些 USB-RS232 转接器，不能兼容，可让宏晶帮你购买经过测试的转换器。
Step4/ 步骤 4：选择下次冷启动后，时钟源为“内部 R/C 振荡器”还是“外部晶体或时钟”。
Step5/ 步骤 5：选择“Download/ 下载”按钮下载用户的程序进单片机内部，可重复执行
Step5/ 步骤 5， 也可选择“Re-Download/ 重复下载”按钮
下载时注意看提示，主要看是否要给单片机上电或复位， 下载速度比一般通用编程器快。
一定要先选择“Download/ 下载”按钮，然后再给单片机上电复位(先彻底断电)，而不要先上电，先上电，检测不到合法的下载命令流，单片机就直接跑用户程序了。

关于硬件连接：

(1). MCU/ 单片机 RXD(P3.0) --- RS-232 转换器 --- PC/ 电脑 TXD(COM Port Pin3)
(2). MCU/ 单片机 TXD(P3.1) --- RS-232 转换器 --- PC/ 电脑 RXD(COM Port Pin2)
(3). MCU/ 单片机 GND ---------------------- PC/ 电脑 GND(COM Port Pin5)
(4). 如果您的系统 P3.0/P3.1 连接到 RS-485 电路，推荐
在选项里选则“下次冷启动需要 P1.0/P1.1 = 0,0 才判 P3.0/RxD 有无合法下载命令流”
这样冷启动后如 P1.0, P1.1 不同时 0,单片机直接运行用户程序，免得由于 RS-485 总线上的乱码造成单片机反复判断乱码是否为合法，浪费几百 mS 的时间

(5). RS-232 转换器可选用 STC232/MAX232/SP232(4.5-5.5V),STC3232/MAX3232/SP3232(3V-5.5V).
STC232/MAX232/SP232 尽量选用 SOP 封装(窄体)，STC3232 尽量选用 SOP 封装(窄体).

如用户系统没有 RS-232 接口，
可使用 STC-ISP Ver 3.0A.PCB 演示板作为编程工具

STC-ISP Ver 3.0APCB 板如焊接的是 STC12C5410AD 的线路，则
可完成 STC12C5410AD 系列单片机的 ISP 下载编程 / 烧录用户程序的功能。

在 STC-ISP Ver 3.0A PCB 板完成下载 / 烧录：

关于硬件连接：

(1.) 根据单片机的工作电压选择单片机电源电压
  A. 5V 单片机,短接 JP1 的 MCU-VCC, +5V 电源管脚
  B. 3V 单片机,短接 JP1 的 MCU-VCC, 3.3V 电源管脚

(2.)连接线(宏晶提供)
  A. 将一端有 9 芯连接座的插头插入 PC/ 电脑 RS-232 串行接口插座用于通信
  B. 将同一端的 USB 插头插入 PC/ 电脑 USB 接口用于取电
  C. 将只有一个 USB 插头的一端插入宏晶的 STC-ISP Ver 3.0A PCB 板 USB1 插座用于 RS-232 通信和供电,此时 USB +5V Power 灯亮(D43,USB 接口有电)

(3.)其他插座不需连接

(4.)SW1 开关处于非按下状态，此时 MCU-VCC Power 灯不亮(D41)， 没有给单片机通电

(5.)SW3 开关
  处于非按下状态，P1.0, P1.1 = 1, 1,不短接到地。
  处于按下状态， P1.0, P1.1 = 0, 0,短接到地。
  如果单片机已被设成“下次冷启动 P1.0/P1.1 = 0,0 才判 P3.0/RxD 有无合法下载命令流”
  就必须将 SW3 开关处于按下状态，让单片机的 P1.0/P1.1 短接到地

(6.)将单片机插进 U1-Socket 锁紧座，锁紧单片机,注意单片机是 20-PIN,而 U1-Socket 锁紧座是 40-PIN,我们的设计是靠下插，靠近晶体的那一端插。

(7.)关于软件：选择“Download/ 下载”(必须在给单片机上电之前让 PC 先发一串合法下载命令)

(8.)按下 SW1 开关，给单片机上电复位， 此时 MCU-VCC Power 灯亮(D41)
  此时 STC 单片机进入 ISP 模式(STC12C5410AD 系列冷启动进入 ISP)

(9.)下载成功后，再按 SW1 开关，此时 SW1 开关处于非按下状态，MCU-VCC Power 灯不亮(D41)，给单片机断电，取下单片机。

利用 STC-ISP Ver 3.0A PCB 板进行 RS-232 转换单片机在用户自己的板上完成下载 / 烧录：

1. U1-Socket 锁紧座不得插入单片机
2. 将用户系统上的电源(MCU-VCC,GND)及单片机的 P3.0/RXD,P3.1/TXD 接入转换板 CN2 插座
   这样用户系统上的单片机就具备了与 PC/ 电脑进行通信的能力
3. 将用户系统的单片机的 P1.0,P1.1 接入转换板 CN2 插座(如果需要的话)
4. 如须 P1.0, P1.1 = 0, 0,短接到地，可在用户系统上将其短接到地，或将 P1.0/P1.1 也从用户系统上引到 STC-ISP Ver3.0A PCB 板上，将 SW3 开关按下，则 P1.0/P1.1=0,0。
5. 关于软件：选择“Download/ 下载”
6. 给单片机系统上电复位(注意是从用户系统自供电，不要从电脑 USB 取电,电脑 USB 座不插)
7. 下载程序时，如用户板有外部看门狗电路，不得启动，单片机必须有正确的复位,但不能在 ISP 下载程序时被外部看门狗复位,如有，可将外部看门狗电路 WDI 端 / 或 WDO 端浮空
8. 如有 RS-485 晶片连到 P3.0/Rxd,P3.1/Txd,或其他线路，在下载时应将其断开。

STC12C5410AD 系列编译器 / 汇编器，编程器，仿真器

STC 单片机应使用何种编译器 / 汇编器：

1.任何老的编译器 / 汇编器都可以支持，流行用 Keil C51
2.把 STC 单片机，当成 Intel 的 8052/87C52/87C54/87C58 就可以了
3.如果要用到扩展的专用特殊功能寄存器，直接对该地址单元设置就行了，当然先声明特殊功能寄存器的地址较好

编程烧录器:

我们有: STC12C5410AD 系列 ISP 经济型下载编程工具(人民币 50 元，可申请免费样品)
注意:有专门下载 28PIN/20PIN 的不同演示板,
28PIN 是 28PIN 的演示板,20PIN 是 20PIN 的演示板

仿真器:如您已有老的仿真器，可仿真普通 8052 的基本功能

STC12C5410AD 系列单片机扩展功能如它仿不了
可以用 STC-ISP 直接下载用户程序看运行结果就可以了
无须添加新的设备

内部数据 RAM 存储器

内部数据 RAM 存储器

STC12C5410AD 系列单片机内部有 256 字节常规的 RAM,256 字节的扩展 RAM

器件的内部常规数据存储器由 3 部分组成：

1. 低 128 字节 RAM（00H～7FH)，可直接和间接寻址,用“MOV”和“MOV @Ri”
2. 高 128 字节 RAM（80H～FFH)，间接寻址，用“MOV @Ri”
3. 特殊功能寄存器（80H～FFH)，只可直接寻址，　用“MOV”

由于高 128 字节和 SFR（特殊功能寄存器）占用相同的地址，因此高 128 字节 RAM 空间必须用间接寻址（MOV @Ri）来区分。特殊功能寄存器（80H～FFH)，只可直接寻址（用“MOV”）来区分。尽管 RAM 和 SFR 的地址相同，但它们在物理上是独立的。

扩展数据 RAM

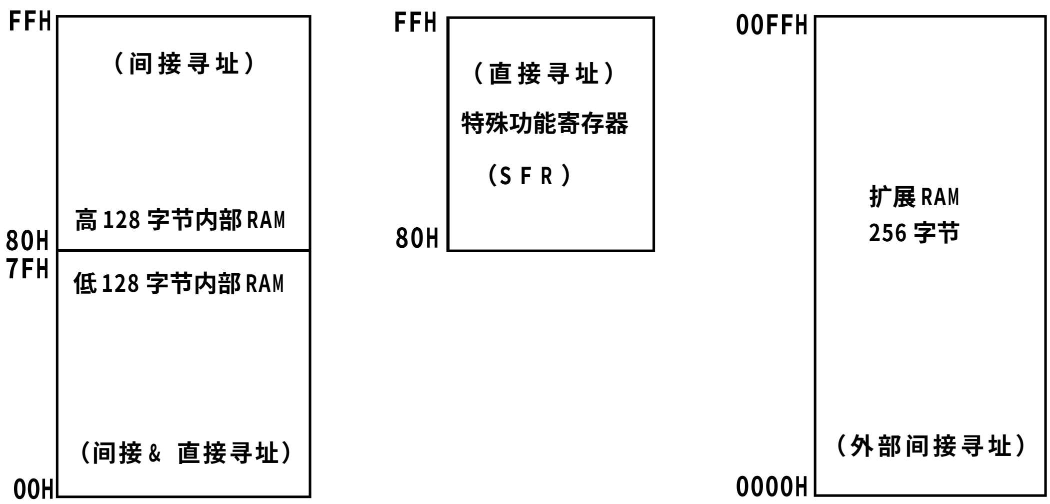
STC12C5410AD 系列有 256 字节的扩展 RAM，称其为 XRAM（附加 RAM)，用“MOVX”寻址。

扩展的 256 字节 RAM（0000H～00FFH)，通过 MOVX 指令间接寻址。

使用“MOVX @DPTR” / “MOVX @Ri”

C 语言中,可使用 xdata 声明存储类型即可,如:

```
unsigned char xdata  i = 0;
```



内部常规 256 字节 RAM 间接寻址测试程序

```
TEST_CONST EQU   5AH
;TEST_RAM  EQU   03H

     ORG   0000H
     LJMP  INITIAL

     ORG   0050H
INITIAL:


     MOV   R0,   #253

     MOV   R1,   #3H
TEST_ALL_RAM:
     MOV   R2,   #0FFH
TEST_ONE_RAM:
     MOV   A,    R2
     MOV   @R1,  A
     CLR   A
     MOV   A,    @R1

     CJNE  A,    2H,   ERROR_DISPLAY
     DJNZ  R2,   TEST_ONE_RAM
     INC   R1
     DJNZ  R0,   TEST_ALL_RAM

OK_DISPLAY:
     MOV   P1,   #11111110B
Wait1:
     SJMP  Wait1

ERROR_DISPLAY:
     MOV   A,    R1
     MOV   P1,   A
Wait2:
     SJMP  Wait2

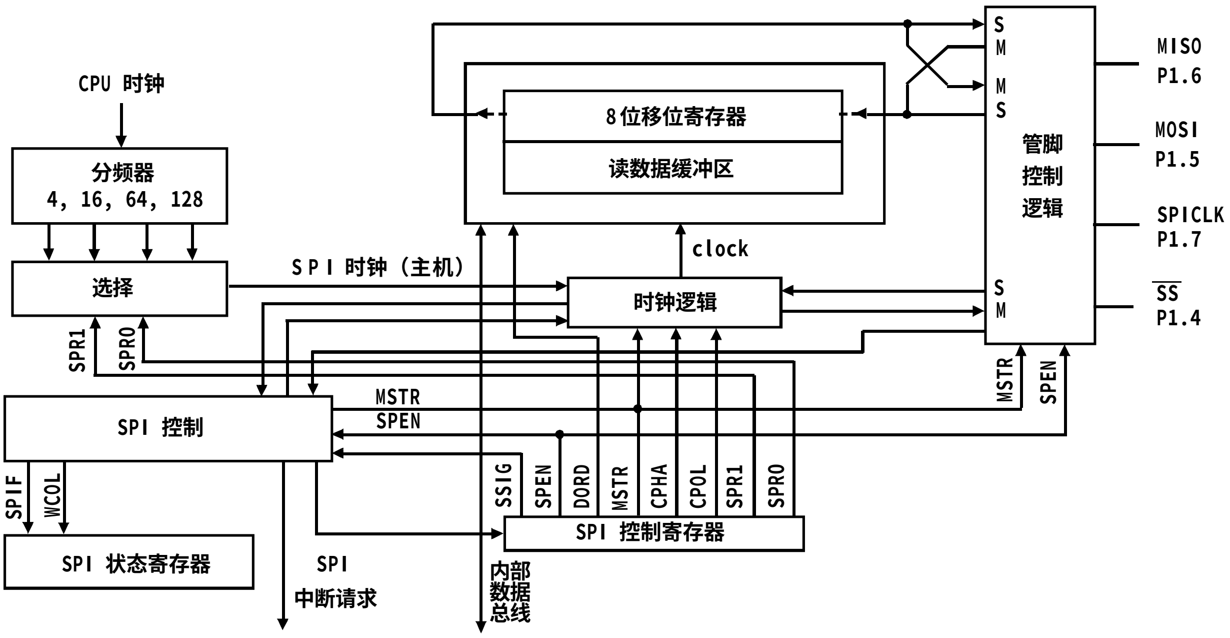
     END
```

串行外围接口（SPI）

STC12C5410AD 系列单片机还提供另一种高速串行通信接口——SPI 接口。SPI 是一种全双工、高速、同步的通信总线，有两种操作模式：主模式和从模式。在主模式中支持高达 3Mbit/s 的速率(工作频率为 12MHz 时,如果 CPU 主频采用 20MHz 到 36MHz,则可更高，从模式时速度无法太快，Fosc/8 以内较好)，还具有传输完成标志和写冲突标志保护。

STC12C5410AD 系列 1T 8051 单片机 SPI 功能模块特殊功能寄存器　SPI Management SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
SPCTL	85h	SPI Control Register	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	0000,0000
SPSTAT	84h	SPI Status Register	SPIF	WCOL	-	-	-	-	-	-	00xx,xxxx
SPDAT	86h	SPI Data Register									0000,0000



SPI 方框图

SPI 接口有 4 个管脚：SPICLK/P1.7, MOSI/P1.5, MISO/P1.6 和 $\overline{SS}$/P1.4。

● SPICLK, MOSI 和 MISO 通常和两个或更多 SPI 器件连接在一起。数据通过 MOSI 由主机传送到从机，通过 MISO

由从机传送到主机。SPICLK 信号在主模式时为输出，在从模式时为输入。如果 SPI 系统被禁止，即 SPEN (SPCTL.6)=0(复位值)，这些管脚都可作为 I/O 口使用。

● /SS 为从机选择管脚。在典型的配置中，SPI 主机使用 I/O 口选择一个 SPI 器件作为当前的从机。

SPI 从器件通过其 /SS 脚确定是否被选择。如果满足下面的条件之一，/SS 就被忽略：

—如果 SPI 系统被禁止，即 SPEN(SPCTL.6)=0(复位值)

—如果 SPI 配置为主机，即 MSTR(SPCTL.4)=1，并且 P1.4 配置为输出（通过 P1M0.4 和 P1M1.4）

—如果 /SS 脚被忽略，即 SSIG(SPCTL.7)位＝1，该脚配置用于 I/O 口功能。

注：即使 SPI 被配置为主机（MSTR＝1），它仍然可以通过拉低 /SS 脚配置为从机（如果 P1.4 配置为输入且 SSIG=0)。要使能该特性，应当置位 SPIF(SPSTAT.7)。

典型连接如 SPI 图 1～3 所示。

SPI 控制寄存器的位分配（SPCTL- 地址：85h）

位	7	6	5	4	3	2	1	0
符 号	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
复 位	0	0	0	0	0	1	0	0

SPI 控制寄存器的位描述（SPCTL – 地址：85h）

位	符号	描 述
0	SPR0	SPR0/SPR1是SPI 时钟速率选择控制位。
1	SPR1	SPR1, SPR0： 0 0 —CPU_CLK/4 0 1 —CPU_CLK/16 1 0 —CPU_CLK/64 1 1 —CPU_CLK/128
2	CPHA	SPI 时钟相位选择（见SPI图4～图7）： 1：数据在SPICLK 的前时钟沿驱动，并在后时钟沿采样。 0：数据在/SS 为低（SSIG＝00）时被驱动，在SPICLK 的后时钟沿被改变，并在前时钟沿被采样。 （注：SSIG=1 时的操作未定义）
3	CPOL	SPI 时钟极性（见SPI图4～图7）： 1：SPICLK 空闲时为高电平。SPICLK 的前时钟沿为下降沿而后沿为上升沿。 0：SPICLK 空闲时为低电平。SPICLK 的前时钟沿为上升沿而后沿为下降沿。
4	MSTR	主/从模式选择（见SPI 主从选择表）。
5	DORD	SPI 数据顺序： 1：数据字的LSB(最低位) 最先发送；0：数据字的MSB(最高位) 最先发送。
3	SPEN	SPI 使能。 1：SPI 使能。 0：SPI 被禁止，所有SPI 管脚都作为I/O 口使用。
7	SSIG	/SS 忽略。 1：MSTR（位4）确定器件为主机还是从机。 0：/SS 脚用于确定器件为主机还是从机。/SS 脚可作为I/O 口使用（见SPI 主从选择表）。

SPI 状态寄存器的位分配（SPSTAT – 地址：84h）

位	7	6	5	4	3	2	1	0
符号	SPIF	WCOL	-	-	-	-	-	-
复位	0	0	X	X	X	X	X	X

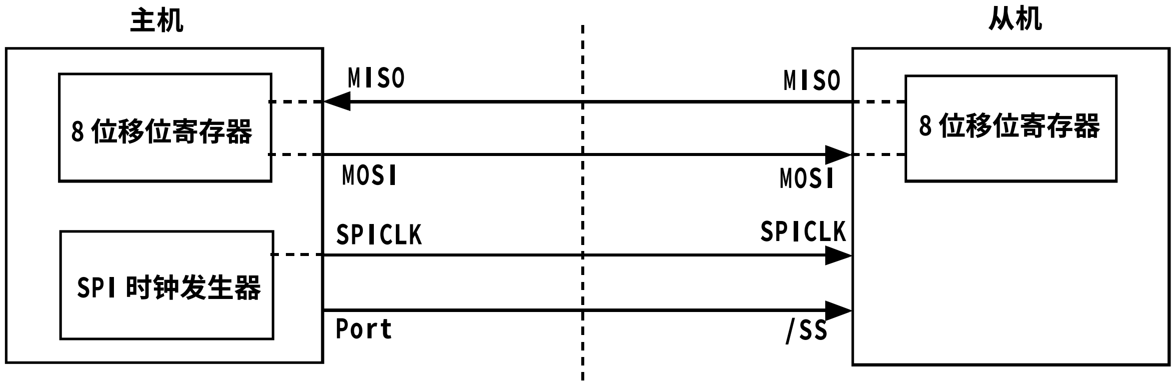
SPI 状态寄存器的位描述（SPSTAT – 地址：84h）

位	符号	符号
7	SPIF	SPI 传输完成标志。当一次串行传输完成时，SPIF 置位，并当ESPI 和EA 都置位时产生中断。当SPI 处于主模式且SSIG=0 时，如果/SS 为输入并被驱动为低电平，SPIF 也将置位。SPIF标志通过软件向其写入“1”清零。
6	WCOL	SPI 写冲突标志。在数据传输的过程中如果对SPI 数据寄存器SPDAT 执行写操作，WCOL 将置位。WCOL 标志通过软件向其写入“1”清零。
5 - 0	-	保留

SPI 数据寄存器的位分配（SPDAT – 地址：86h）

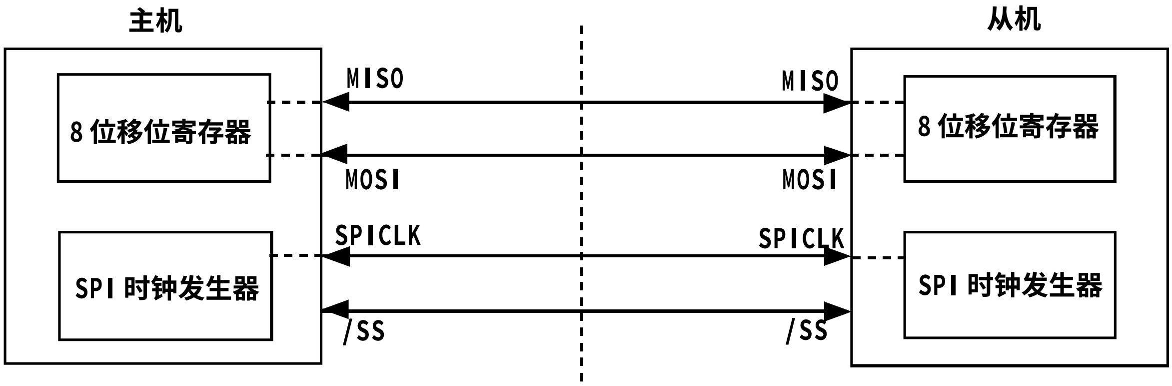
位	7	6	5	4	3	2	1	0
符 号	MSB							LSB
复 位	0	0	0	0	0	0	0	0

SPDAT.7 - SPDAT.0: 传输的数据位Bit7～Bit0



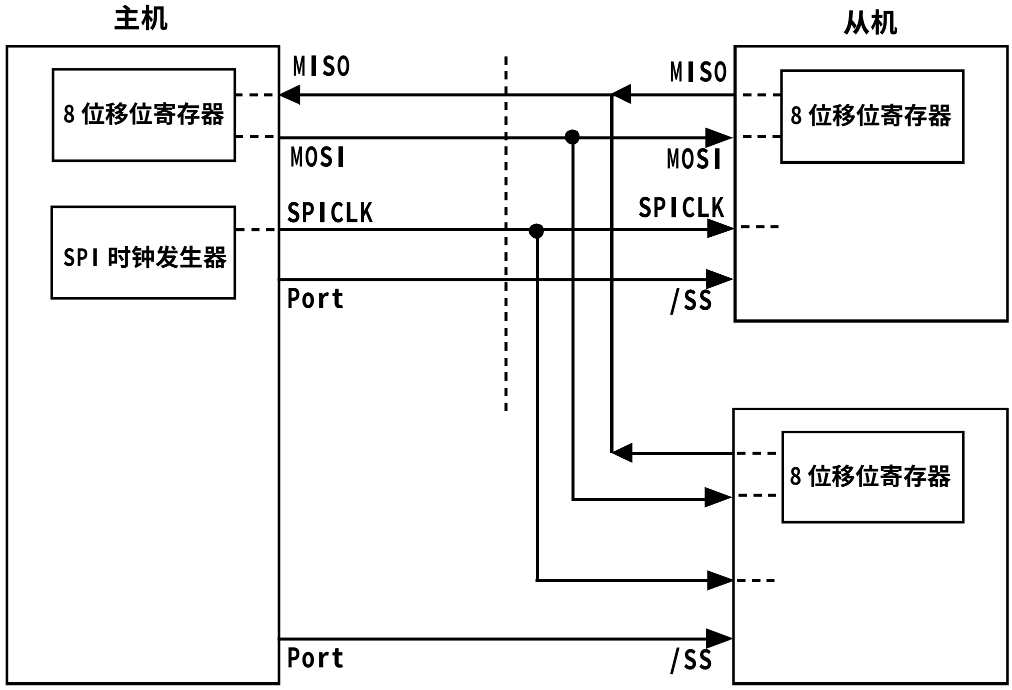
SPI 图 1 SPI 单主机 - 单从机 配置

在上图SPI 图1 中，从机的SSIG(SPCTL.7)为0，/SS 用于选择从机。SPI 主机可使用任何端口（包括P1.4/$\overline{SS}$）来驱动 /SS 脚。



SPI 图 2 SPI 双器件配置（可器件可互为主从）

上图SPI 图2 所示为两个器件互为主从的情况。当没有发生 SPI 操作时，两个器件都可配置为主机(MSTR=1)，将 SSIG 清零并将 P1.4(/SS)配置为准双向模式。当其中一个器件启动传输时，它可将 P1.4 配置为输出并驱动为低电平，这样就强制另一个器件变为从机。



SPI 图 3 SPI 单主机 - 多从机 配置

在上图SPI 图3 中，从机的SSIG(SPCTL.7)为0，从机通过对应的 /SS 信号被选中。SPI 主机可使用任何端口（包括P1.4/$\overline{SS}$）来驱动 /SS 脚。

对SPI 进行配置

下表 所示为主/从模式的配置以及模式的使用和传输方向。

SPI 主从模式选择

SPEN	SSIG	/SS 脚 P1.4	MSTR	主或从模式	MISO P1.6	MOSI P1.5	SPICLK P1.7	备注
0	X	P1.4	X	SPI 功能禁止	P1.6	P1.5	P1.7	SPI 禁止。P1.4/P1.5/P1.6/P1.7作为普通I/O口使用
1	0	0	0	从机模式	输出	输入	输入	选择作为从机
1	0	1	0	从机模式未被选中	高阻	输入	输入	未被选中。MISO 为高阻状态，以避免总线冲突
1	0	0	1—>0	从机模式	输出	输入	输入	P1.4/ SS 配置为输入或准双向口。SSIG 为0。如果择/SS 被驱动为低电平，则被选择作为从机。当SS 变为低电平时，MSTR将清零。注：当/SS处于输入模式时，如被驱动为低电平且SSIG=0 时，MSTR 位自动清零。
1	0	1	1	主(空闲)	输入	高阻	高阻	当主机空闲时MOSI 和SPICLK 为高阻态以避免总线冲突。用户必须将SPICLK 上拉或下拉（根据CPOL-SPCTL.3 的取值）以避免SPICLK出现悬浮状态。
				主(激活)		输出	输出	作为主机激活时，MOSI 和SPICLK 为推挽输出
1	1	P1.4	0	从	输出	输入	输入	
1	1	P1.4	1	主	输入	输出	输出	

作为从机时的额外注意事项

当CPHA＝0 时，SSIG 必须为0，/SS 脚必须取反并且在每个连续的串行字节之间重新设置为高电平。如果SPDAT 寄存器在/SS 有效（低电平）时执行写操作，那么将导致一个写冲突错误。CPHA=0 且SSIG=0 时的操作未定义。

当CPHA＝1 时，SSIG 可以置位。如果SSIG＝0，/SS 脚可在连续传输之间保持低有效（即一直固定为低电平）。这种方式有时适用于具有单固定主机和单从机驱动MISO 数据线的系统。

作为主机时的额外注意事项

在SPI 中，传输总是由主机启动的。如果SPI 使能（SPEN=1）并选择作为主机，主机对SPI 数据寄存器的写操作将启动SPI 时钟发生器和数据的传输。在数据写入SPDAT 之后的半个到一个SPI 位时间后，数据将出现在MOSI 脚。

需要注意的是，主机可以通过将对应器件的/SS 脚驱动为低电平实现与之通信。写入主机SPDAT 寄存器的数据从MOSI 脚移出发送到从机的MOSI 脚。同时从机SPDAT 寄存器的数据从MISO 脚移出发送到主机的MISO 脚。

传输完一个字节后，SPI 时钟发生器停止，传输完成标志（SPIF）置位并产生一个中断（如果SPI 中断使能）。主机和从机CPU 的两个移位寄存器可以看作是一个16 循环移位寄存器。当数据从主机移位传送到从机的同时，数据也以相反的方向移入。这意味着在一个移位周期中，主机和从机的数据相互交换。

通过/SS 改变模式

如果SPEN=1，SSIG=0且MSTR=1，SPI使能为主机模式。/SS脚可配置为输入或准双向模式。这种情况下，另外一个主机可将该脚驱动为低电平，从而将该器件选择为SPI 从机并向其发送数据。

为了避免争夺总线，SPI系统执行以下动作:

1)MSTR清零并且CPU变成从机。这样SPI就变成从机。MOSI 和SPICLK强制变为输入模式，而MISO则变为输出模式。

2）SPSTAT的SPIF标志位置位。如果SPI中断已被使能，则产生SPI中断。

用户软件必须一直对MSTR位进行检测，如果该位被一个从机选择所清零而用户想继续将SPI作为主机，这时就必须重新置位MSTR，否则就进入从机模式。

写冲突

SPI在发送时为单缓冲，在接收时为双缓冲。这样在前一次发送尚未完成之前，不能将新的数据写入移位寄存器。当发送过程中对数据寄存器进行写操作时，WCOL位（SPSTAT.6）将置位以指示数据冲突。在这种情况下，当前发送的数据继续发送，而新写入的数据将丢失。

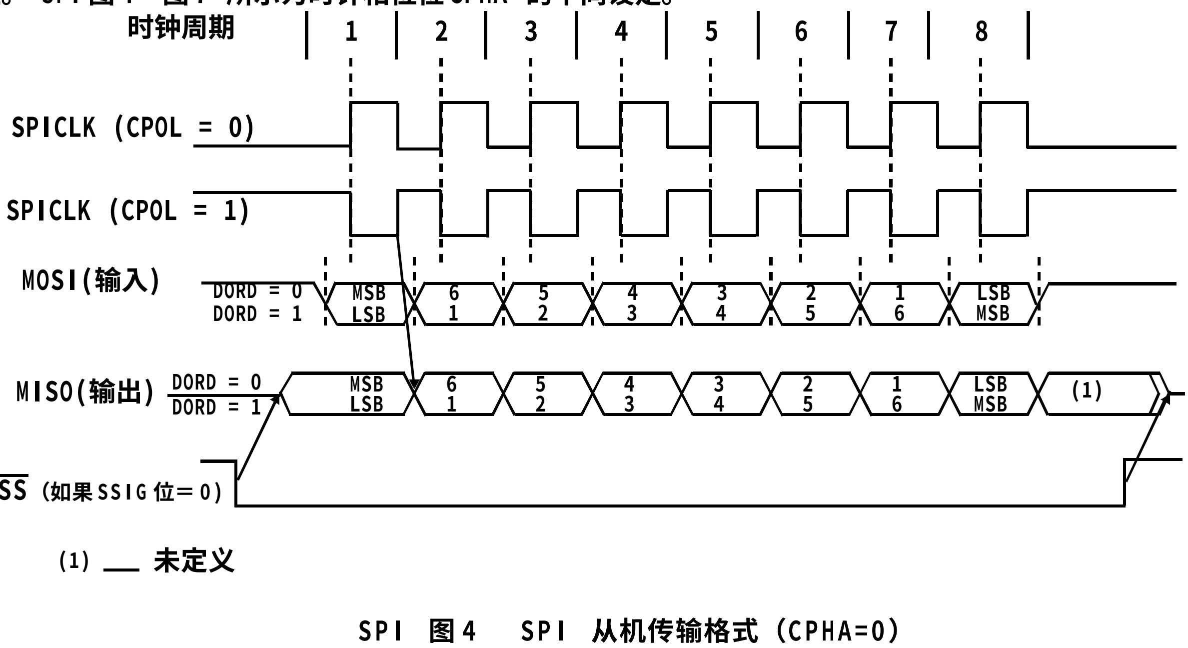
当对主机或从机进行写冲突检测时，主机发生写冲突的情况是很罕见的，因为主机拥有数据传输的完全控制权。但从机有可能发生写冲突，因为当主机启动传输时，从机无法进行控制。

接收数据时，接收到的数据传送到一个并行读数据缓冲区，这样将释放移位寄存器以进行下一个数据的接收。但必须在下个字符完全移入之前从数据寄存器中读出接收到的数据，否则，前一个接收数据将丢失。

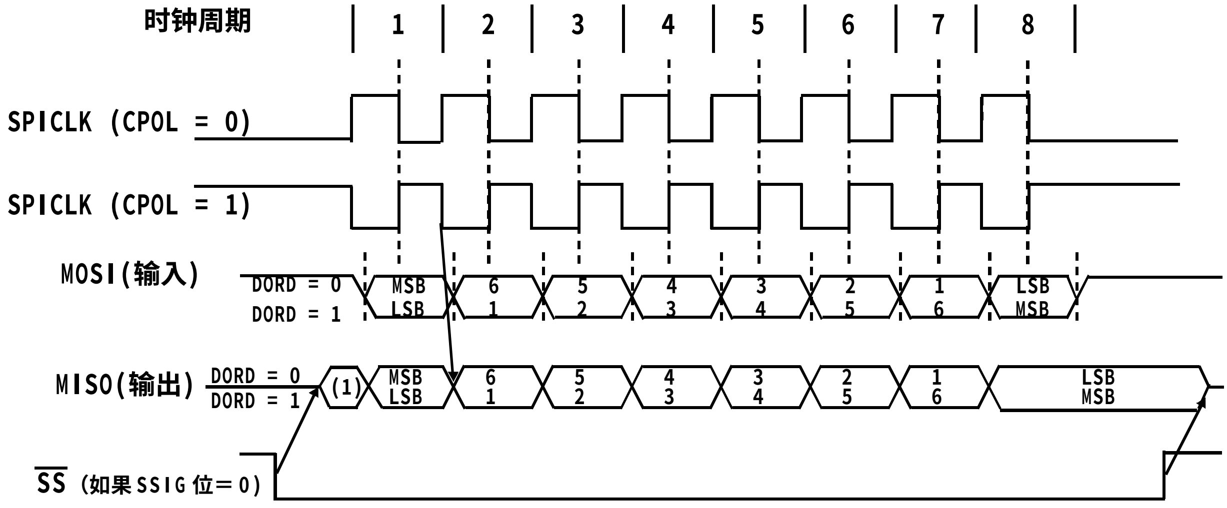
WCOL可通过软件向其写入“1”清零。

数据模式

时钟相位位(CPHA)允许用户设置采样和改变数据的时钟边沿。时钟极性位CPOL允许用户设置时钟极性。 SPI图4～图7 所示为时钟相位位CPHA 的不同设定。

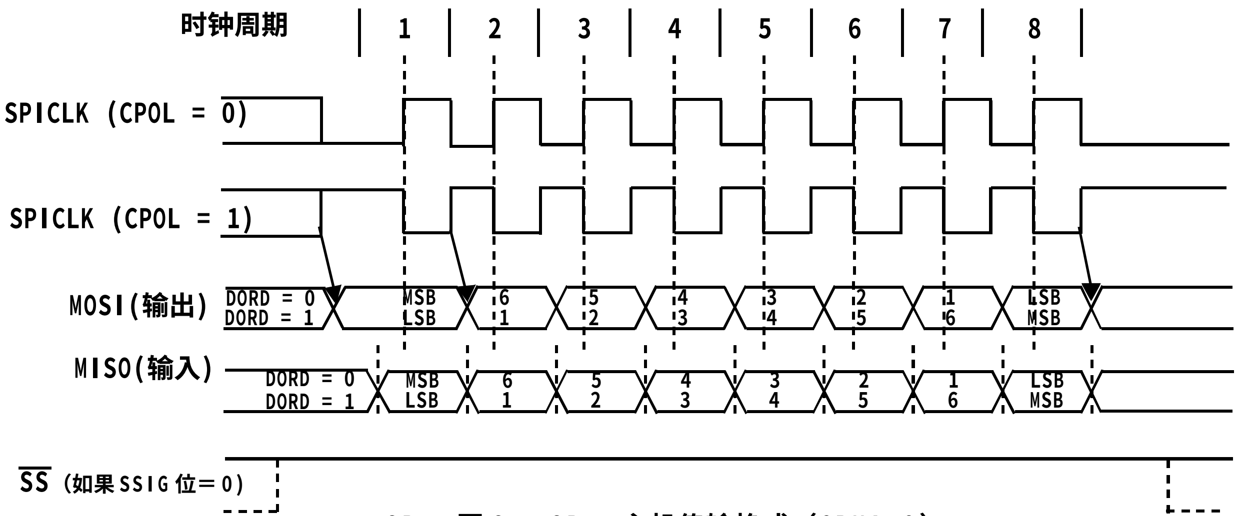


SPI 图4 SPI 从机传输格式（CPHA=0）

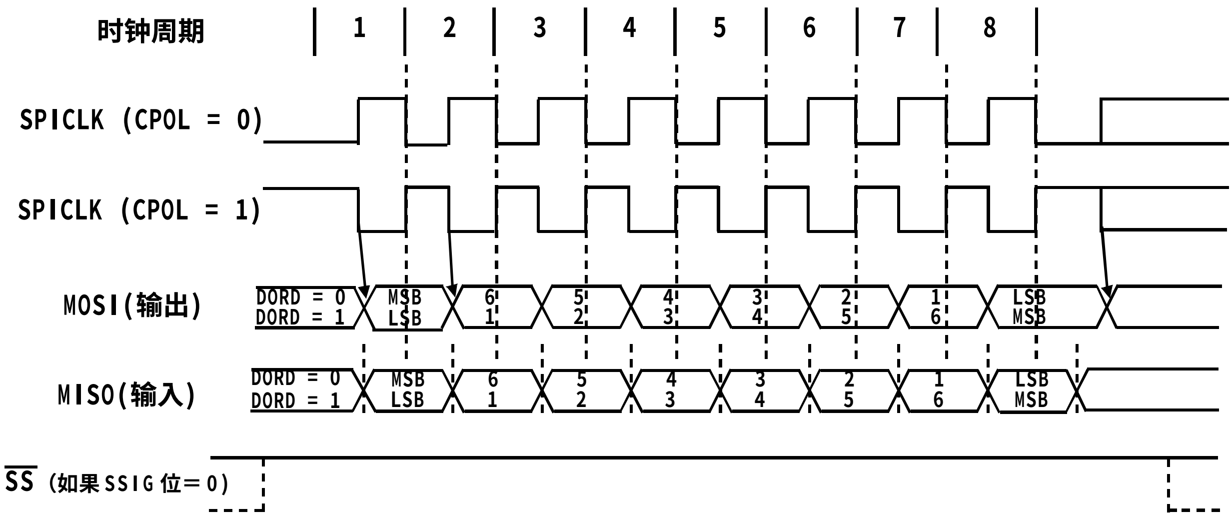


(1) ___ 未定义

SPI 图5 SPI 从机传输格式（CPHA=1）



SPI 图6 SPI 主机传输格式（CPHA=0）



SPI 图7 SPI 主机传输格式（CPHA=1）

SPI 时钟预分频器选择

SPI 时钟预分频器选择是通过SPCTL 寄存器中的SPR1-SPR0 位实现的

SPI 功能示例程序(适用于单主单从系统)

```
;/* ---  STC International Limited ---------------- */
;/* ---  宏晶科技    姚永平    2006/1/6    V1.0 ---- */
;/* ---  one_master_one_slave --------------------- */
;/* ---  STC12C5412AD, STC12C5410AD, STC12C5408AD ------- */
;/* ---  STC12C5406AD, STC12C5404AD, STC12C5402AD ------- */
;/* ---  STC12C5052AD, STC12C4052AD, STC12C3052AD ------- */
;/* ---  STC12C2052AD, STC12C1052AD, STC12C0552AD ------- */
;/* ---  Mobile: 13922805190 ---------------------- */
;/* ---  Fax: 0755-82944243 ----------------------- */
;/* ---  Tel: 0755-82948409 ----------------------- */
;/* ---  Web: www.mcu-memory.com ------------------ */

;-------------------------------------------------------------------------------
;1. 本示例程序演示 STC12C2052AD 和 STC12C5410AD 系列 MCU 的 SPI 功能，适用于
;    单主单从系统
;-------------------------------------------------------------------------------
;2. 硬件连接:三线连接
;
;          主单片机       I/O口                    I/O口     从单片机
;
;      +--------------+  MISO    <-- 位流方向     MISO +--------------+
;      |     SPI      |<<----------------------------|      SPI      |
;      |8位移位寄存器 |                              |8位移位寄存器  |
;      |              |---------------------------->>|               |
;      +-------+------+  MOSI     位流方向 -->     MOSI +-------^------+
;              |                                              |
;              |         SCLK                    SCLK         |
;              +------------------->>---------------------+
;
;     除此之外，主单片机的 RS-232 串行口通过 RS-232 转换器与 PC 机的 RS-232
; 串行口相连接。
;
;3. SPI 通讯过程：
;     主单片机与从单片机的 SPI 8位移位寄存器连接成一个循环的 16 位移位寄存器。
;当主单片机程序向 SPDAT 写入一个字节时，立即启动一个连续的 8 位移位通讯过程:
;主单片机的 SCLK 脚向从单片机的 SCLK 脚发出一串脉冲，在这串脉冲的驱动下，主
;单片机 SPI 8位移位寄存器中的数据移到了从单片机的 SPI 8位移位寄存器中；与此
;同时，从单片机 SPI 8位移位寄存器中的数据移到了主单片机的 SPI 8位移位寄存器
; 中。利用这样的数据交换机制，主单片机既可向从单片机发送数据，又可读从单片机
; 中的数据。
;
;4. 使用方法
;   a) 修改程序，使 MASTER EQU 1 的那行有效。汇编后的程序代码下载到主单片机中。
;   b) 修改程序，使 MASTER EQU 0 的那行有效。汇编后的程序代码下载到从单片机中。
```

```
;   c） 给主、从单片机上电。
;   d）用串口调试助手(STC 的 ISP 下载程序 STC-ISP.exe 3.2 以上版本提供了该功能)
;       向主单片机发送一串数据。
;           主单片机的 RS-232 串口每收到一个字节就立刻将收到的字节通过 SPI 口
;       发送到从单片机中，与此同时主单片机会收到从单片机发回的一个字节(见3. SPI
;       通讯过程)，主单片机又立刻把这个字节通过 RS-232 口发送到 PC 机。
;           从单片机的 SPI 口收到的数据后，把收到的数据放到自己的 SPDAT 寄存器
;       中，当下一次主单片机发送一个字节时把数据发回到主单片机。
;   e） 在串口调试助手接收区观察接收的数据。
;
;5. 怎样用巡测方式接收 SPI 数据
;       本示例为中断方式接收 SPI 口数据，若想用巡测方式接收 SPI 数据可以用以下
;    几行指令实现：

;   Wait_SPI_Receive_Byte:
;       MOV A, SPSTAT                       ;判收到从 SPI 发回的数据否
;       ANL A, #80H
;       JZ  Wait_SPI_Receive_Byte           ;SPI 未收到数据，继续等待
;       MOV   A, SPDAT                      ;SPI 已收到数据，将收到的数据送累加器 A
;         ...
;
;6. 实验条件: MCU 晶振频率 Fosc = 18.432MHz, PC 机 RS232 串口波特率等于 57600
;    实验结果: SPI 口传输数据无误。
;        由于本程序的 RS232 接收，SPI 端口的接收都没有使用接收缓冲区，所以 RS232
;    串口波特率不要高于 57600，若使用接收缓冲区，波特率可以到 115200 以上。
;-------------------------------------------------------------------------------
;定义常量
;-----------------
;定义功能常量，以下两行注释其中一行，取消另一行注释使之有效
;MASTER    EQU 1            ;汇编后的程序代码下载到主单片机中
MASTER    EQU 0            ;汇编后的程序代码下载到从单片机中
;-----------------
;定义波特率自动重装数常量
;以下波特率是 PCON.7 = 0 时的数值，若使 PCON.7 = 1 可将波特率加倍
;RELOAD_8BIT_DATA  EQU 0FFH   ;Fosc=22.1184MHz, Baud = 57600
;RELOAD_8BIT_DATA  EQU 0FBH   ;Fosc=18.432MHz, Baud=9600, 1T 运行时 Baud=115200
RELOAD_8BIT_DATA   EQU 0F6H   ;Fosc=18.432MHz, Baud=4800, 1T 运行时 Baud=57600
;RELOAD_8BIT_DATA  EQU 0FFH   ;Fosc=11.059MHz, Baud = 28800、
;-------------------------------------------------------------------------------
;定义特殊功能寄存器
AUXR     EQU 8EH
;AUXR 特殊功能寄存器的 bit3 是 SPI 中断允许控制位 ESPI
;IE 特殊功能寄存器的 bit5 是ADC和SPI两个中断共享的总中断允许控制位 EADC_SPI
;要产生 SPI 中断，需要 ESPI/EADC_SPI/EA 都为 1
;-------------------------------------------------------------------------------
```

```
;定义 SPI 特殊功能寄存器，详细说明见本程序的后部或 STC 12C5410AD 中文指南
SPCTL    EQU 85H
SPSTAT   EQU 84H
SPDAT    EQU 86H

EADC_SPI EQU IE.5
;------------------------------------------------------------------------------
;定义 SPI 脚
SCLK     EQU P1.7
MISO     EQU P1.6
MOSI     EQU P1.5
SS       EQU P1.4
;------------------------------------------------------------------------------
;定义单片机管脚
LED_MCU_START       EQU P3.4
;------------------------------------------------------------------------------
;定义变量
Flags        EQU 20H
SPI_Receive EQU Flags.0             ;SPI 端口收到数据标志位
SPI_buffer  EQU 30H                 ;该变量用于保存 SPI 端口收到的数据
;------------------------------------------------------------------------------
    ORG   0000H
    LJMP  MAIN
;------------------------------------------------------------------------------
    ORG   002BH                     ;ADC_SPI 中断服务程序入口
    LJMP  ADC_SPI_Interrupt_Routine
;------------------------------------------------------------------------------
    ORG   0080H
MAIN:
    CLR   LED_MCU_START             ;点亮 MCU 开始工作指示灯
    MOV   SP, #7FH
    ACALL Init_System               ;系统初始化
if MASTER
Check_RS232:
        JNB   RI, Master_Check_SPI  ;判 RS-232 串口中收到数据否
        ;主单片机 RS-232 串口已收到新的数据
        ACALL Get_Byte_From_RS232   ;主单片机将RS-232串口中收到的数据送到累加器 A
        ACALL SPI_Send_Byte         ;主单片机将累加器 A 中的数据发送到从机 SPI
        SJMP  Check_RS232
Master_Check_SPI:
        JNB   SPI_Receive, Check_RS232 ;判收到从 SPI 发回的数据否
        ;主单片机 SPI 端口已收到新的数据
        MOV   A, SPI_buffer         ;将 "从 SPI 发回的数据" 送到累加器 A
        CLR   SPI_Receive           ;清0 主单片机 SPI 端口收到数据标志位
        ACALL RS232_Send_Byte       ;将累加器 A 中的数据发送到 PC 机
        SJMP  Check_RS232
```

```
else
Slave_Check_SPI:
        JNB    SPI_Receive, Slave_Check_SPI ;判收到主 SPI 发回的数据否
        ;从单片机 SPI 端口已收到新的数据
        MOV    A, SPI_buffer            ;取 "主单片机 SPI 端口发的数据"
        CLR    SPI_Receive              ;清0 从单片机 SPI 端口收到数据标志位
        MOV    SPDAT, A                 ;将收到数据送 SPDAT，准备下一次通讯时发回
        SJMP   Slave_Check_SPI
endif
;-------------------------------------------------------------------------------
ADC_SPI_Interrupt_Routine:              ;ADC_SPI 中断服务程序
    ;SPI 中断服务程序
    MOV    SPSTAT, #11000000B           ;0C0H，清0 标志位 SPIF 和 WCOL
                    ;特别注意：是向标志位 SPIF/WCOL 写1，将 SPIF/WCOL 清成0
                    ;特别注意：不是向标志位 SPIF/WCOL 写0，将 SPIF/WCOL 清成0
    MOV    A, SPDAT                     ;保存收到的数据
    MOV    SPI_buffer, A
    SETB   SPI_Receive                  ;树立 SPI 端口收到数据标志
    RETI
;-------------------------------------------------------------------------------
Init_System:
    ACALL  Initial_UART                 ;初始化串口
    ACALL  Initial_SPI                  ;初始化 SPI
    MOV    Flags, #0                    ;清标志字
    SETB   EA                           ;开总中断
    RET
;-------------------------------------------------------------------------------
Initial_UART:                           ;初始化串口
; SCON  Bit:   7       6     5     4     3     2     1    0
;            SM0/FE   SM1   SM2   REN   TB8   RB8   TI   RI
    MOV    SCON, #50H                   ;0101,0000 8位可变波特率，无奇偶校验

    MOV    TMOD, #21H                   ;T1 为自动重装模式
    MOV    TH1, #RELOAD_8BIT_DATA
    MOV    TL1, #RELOAD_8BIT_DATA

;     MOV    PCON, #80H                  ;取消本行指令注释，波特率加倍。
;使以下两行有效，波特率快12倍，即波特率 = 4800*12=57600
    MOV    A, #01000000B                ;T1 以 1T 的速度计数，是普通8051的12倍
    ORL    AUXR, A

    SETB   TR1                          ;启动定时器1 开始计数
    RET
;-------------------------------------------------------------------------------
Initial_SPI:                            ;初始化 SPI
;SPI 控制寄存器
```

```
;           7       6       5       4       3       2       1       0
;SPCTL    SSIG    SPEN    DORD    MSTR    CPOL    CPHA    SPR1    SPR0

if MASTER
    MOV     SPCTL,#11111100B            ;0FCH, 忽略 SS 脚, 设为主机
    ;SSIG=1: 忽略SS脚
    ;SPEN=1：允许 SPI 工作
    ;DORD=1：先传低位 LSB
    ;MSTR=1：设为主机

    ;CPOL=1：SPI 空闲时 SPICLK = 1，前跳变沿是下降沿，后跳变沿是上升沿。
    ;CPHA=1：数据由 SPICLK 前跳变沿驱动到 SPI 口线，SPI 模块在后跳变沿采样数据。
    ;SPR1，SPR0 = 00：主模式时 SPI 时钟源选择为 fosc/4
else
    MOV     SPCTL,#11101100B            ;0ECH, 忽略 SS 脚, 设为从机
    ;SSIG=1: 忽略SS脚
    ;SPEN=1：允许 SPI 工作
    ;DORD=1：先传低位 LSB
    ;MSTR=0：设为从机

    ;CPOL=1：SPI 空闲时 SPICLK = 1，前跳变沿是下降沿，后跳变沿是上升沿。
    ;CPHA=1：数据由 SPICLK 前跳变沿驱动到 SPI 口线，SPI 模块在后跳变沿采样数据。
    ;SPR1，SPR0 = 00：主模式时 SPI 时钟源选择为 fosc/4
endif
    MOV     SPSTAT,#11000000B           ;清0 标志位 SPIF(SPSTAT.7), WCOL(SPSTAT.6)
                                        ;向该两个标志位写"1" 会将它们清0
    MOV     A, #00001000B
    ORL     AUXR, A                     ;令ESPI(AUXR.3)=1,允许SPIF(SPSTAT.7)产生中断
    SETB    EADC_SPI                    ;开 ADC 中断和 SPI 中断共享的总中断控制位
    RET
;------------------------------------------------------------------------------
RS232_Send_Byte:                        ;RS232 串口发送一个字节
    CLR     TI                          ;清零串口发送中断标志
    MOV     SBUF, A
RS232_Send_Wait:
    JNB     TI, RS232_Send_Wait         ;等待发送完毕，未发送完毕跳回本行
    CLR     TI                          ;清零串口发送中断标志
    RET
;------------------------------------------------------------------------------
;此段程序只有主 MCU 调用
SPI_Send_Byte:                          ;SPI 发送一个字节
    CLR     EADC_SPI                    ;关 ADC 中断和 SPI 中断共享的总中断控制位
    MOV     SPDAT, A                    ;SPI 发送数据
SPI_Send_Byte_Wait:
    MOV     A, SPSTAT                   ;等待 SPIF=1 即等待 SPI 发送完毕
    ANL     A, #80H
```

```
    JZ      SPI_Send_Byte_Wait
    SETB   EADC_SPI                      ;开 ADC 中断和 SPI 中断共享的总中断控制位
    RET
;---------------------------------------------------------------------------
Get_Byte_From_RS232:                     ;取 RS-232 串口中收到的数据送累加器 A
    MOV    A, SBUF
    CLR    RI
    RET
;---------------------------------------------------------------------------
    END
;---------------------------------------------------------------------------
;更详细的资料可以参阅 STC12C5410AD.pdf (中文使用说明)。
;
;SPI 控制寄存器
;           7      6      5      4      3      2      1      0
;SPCTL    SSIG   SPEN   DORD   MSTR   CPOL   CPHA   SPR1   SPR0
;
;SSIG: 忽略SS脚，如果 SSIG=1，由 MSTR 位决定 SPI 主模式或从模式，
;       如果 SSIG=0，由 SS 脚决定 SPI 主模式或从模式。
;SPEN：SPI使能位。如果 SPEN=0，SPI功能被禁止，SPI 脚用作普通IO口
;DORD：SPI 数据传输顺序。
;       1：先传低位 LSB
;       0：先传高位 MSB
;MSTR：SPI 主/从模式选择位
;CPOL：SPI 时钟信号极性选择位
;        1：SPI 空闲时 SPICLK = 1，前跳变沿是下降沿，后跳变沿是上升沿。
;        0：SPI 空闲时 SPICLK = 0，前跳变沿是上升沿，后跳变沿是下降沿。
;CPHA：SPI 时钟信号相位选择位
;       1：数据由 SPICLK 前跳变沿驱动到 SPI 口线，SPI 模块在后跳变沿采样数据。
;       0：当 SS 脚为低(SSIG=0)时数据被驱动到口线，并且在 SPICLK 后跳变沿数据
;          被改变(被驱动到口线)，在 SPICLK 前跳变沿数据被采样。注意：SSIG = 1
;          时操作未定义。
;SPR1-SPR0：主模式时 SPI 时钟源选择
;       00：fosc/4
;       01：fosc/16
;       10：fosc/64
;       11：fosc/128
;
;    当 CPHA=0，SSIG 必须等于零并且在传输时 SS 脚也必须一直保持为低。当 SS 有效
;(=0)时向 SPDATA 寄存器写数据就会发生写冲突错误，WCOL 标志被置 1。
;    当 CPHA=1，SSIG 可以等于 0 或 1。如果 SSIG=0，SS 脚在连续的传输时为 0(可以
;一直保持为 0)。当系统中只有一个主和一个从 SPI 时，这是首选配置。
;--------------------
;SPI 状态寄存器
;           7      6      5      4      3      2      1      0
;SPSTAT   SPIF   WCOL    -      -      -      -      -      -
```

```
;SPIF：SPI 传输结束标志。当一次传输结束时， SPIF 被置 1，如果 SPI 中断被打开：
;        ESPI(AUXR.3)=1，EADC_SPI(IE.5)=1，EA(IE.7)=1，就引起中断。如果原来 SPI
;        由 SS 脚确定为是主模式(SSIG=0,SS=1)，当 SS 变成 0 时，SPIF 也会被置 1，
;        表示"模式改变"。向 SPIF 位写 1 将该标志清 0。
;WCOL：SPI 写冲突标志。当一个数据还在传输时，又向数据寄存器 SPDAT 写入数据，WCOL
;        就会被被置 1。向 WCOL 位写 1 将该标志清 0。
;-------------------
;SPI 主/从模式选择
;
;SPEN SSIG SS MATR    模式     MISO   MOSI  SPICLK      注释
;  0    X    X   X    禁止SPI   输入   输入   输入      禁止 SPI 功能
;  1    0    0   0      从      输出   输入   输入      被选为从
;  1    0    1   0     未选从   输入   输入   输入      从，但没有被选中
;  1    0    0  1->0    从      输出   输入   输入      由主变为从
;  1    0    1   1      主      输入   输出   输出
;  1    1    X   0      从      输出   输入   输入      从
;  1    1    X   1      主      输入   输出   输出      主
```

SPI 功能示例程序(适用于单主多从系统)

```
;/* ---  STC International Limited ---------------- */
;/* ---  宏晶科技    姚永平    2006/1/6    V1.0 ---- */
;/* ---  one_master_more_slave -------------------- */
;/* ---  STC12C5412AD, STC12C5410AD, STC12C5408AD ------- */
;/* ---  STC12C5406AD, STC12C5404AD, STC12C5402AD ------- */
;/* ---  STC12C5052AD, STC12C4052AD, STC12C3052AD ------- */
;/* ---  STC12C2052AD, STC12C1052AD, STC12C0552AD ------- */
;/* ---  Mobile: 13922805190 ---------------------- */
;/* ---  Fax: 0755-82944243 ----------------------- */
;/* ---  Tel: 0755-82948409 ----------------------- */
;/* ---  Web: www.mcu-memory.com ------------------ */

;-----------------------------------------------------------------------------
;1. 本示例程序演示 STC12C2052AD 和 STC12C5410AD 系列 MCU 的 SPI 功能，适用于
;   单主多从系统
;-----------------------------------------------------------------------------
;2. 硬件连接:
;
;          主单片机                                    从单片机 #1
;      +-------------+    <-- 位流方向               +--------------+
;      |        MISO |<<--------------------*--------| MISO         |
;      |             |    位流方向 -->      |        |              |
;      |        MOSI |------------------*---|------>>| MOSI         |
;      |             |                  |   |        |              |
;      |        SCLK |--------------*---|---|------>>| SCLK         |
;      |             |              |   |   |        | __           |
;      |        P1.2 |--------------|---|---|------>>| SS           |
;      |             |              |   |   |        |              |
;      |             |              |   |   |        +--------------+
;      |             |              |   |   |
;      |             |              |   |   |         从单片机 #2
;      |             |              |   |   |        +--------------+
;      |             |              |   |  <<--------| MISO         |
;      |             |              |   |            |              |
;      |             |              |   +--------->>| MOSI         |
;      |             |              |                |              |
;      |             |              +------------->>| SCLK         |
;      |             |                               | __           |
;      |        P1.3 |----------------------------->>| SS           |
;      |             |                               |              |
;      +-------------+                               +--------------+
;
;       除此之外，主单片机的 RS-232 串行口通过 RS-232 转换器与 PC 机的 RS-232
;串行口相连接。
```

```
;
;3. SPI 通讯过程：
;      主单片机与从单片机的 SPI 8位移位寄存器连接成一个循环的 16 位移位寄存器。
;当主单片机程序向 SPDAT 写入一个字节时，立即启动一个连续的 8 位移位通讯过程:
;主单片机的 SCLK 脚向从单片机的 SCLK 脚发出一串脉冲，在这串脉冲的驱动下，主
;单片机 SPI 8位移位寄存器中的数据移到了从单片机的 SPI 8位移位寄存器中; 与此
;同时，从单片机 SPI 8位移位寄存器中的数据移到了主单片机的 SPI 8位移位寄存器
; 中。利用这样的数据交换机制，主单片机既可向从单片机发送数据，又可读从单片机
; 中的数据。
;
;4. 使用方法
;   a) 修改程序，使 MASTER_SLAVE EQU 0 的那行有效。汇编后的程序代码下载到
;        主单片机中。
;   b) 修改程序，使 MASTER_SLAVE EQU 1 的那行有效。汇编后的程序代码下载到
;        从单片机 #1 中。
;   c) 修改程序，使 MASTER_SLAVE EQU 2 的那行有效。汇编后的程序代码下载到
;        从单片机 #2 中。
;    d) 给主、从单片机上电。
;   e) 主单片机用 Slave1_SS 和 Slave2_SS 口线选择当前选中的从单片机，每一时刻
;        只有一个从单片机被选中。当 Slave1_SS 的 LED 灯亮时，从单片机 #1 被选中;
;        当 Slave2_SS 的 LED 灯亮时，从单片机 #2 被选中。
;            用串口调试助手(STC 的 ISP 下载程序 STC-ISP.exe 3.2 以上版本提供了
;        该功能)向主单片机发送一串数据。主单片机每收到一个字节就立刻将收到的字节
;        通过 SPI 口发送到当前选中的从单片机中。从单片机 #1 将 SPI 口收到的数据
;        再放到自己的 SPDAT 寄存器中，当下一次主单片机发送一个字节时把数据发回到
;        主单片机; 从单片机 #2 将 SPI 口收到的数据加 1 以后再放到自己的 SPDAT
;        寄存器中，当下一次主单片机发送一个字节时把数据发回到主单片机。
;   f) 在串口调试助手接收区观察接收的数据。
;
;5. 用巡测方式接收 SPI 数据
;          本示例为中断方式接收 SPI 口数据，若想用巡测方式接收 SPI 数据可以用以下
;     几行指令实现：

;   Wait_SPI_Receive_Byte:
;        MOV A, SPSTAT                    ;判收到从 SPI 发回的数据?
;        ANL A, #80H
;        JZ  Wait_SPI_Receive_Byte        ;SPI 未收到数据，继续等待
;          ...                            ;SPI 已收到数据
;          ...
;
;6. 实验条件: MCU 晶振频率 Fosc = 18.432MHz, PC 机 RS232 串口波特率等于 57600
;    实验结果: SPI 口传输数据无误。
;---------------------------------------------------------------------------
;定义常量
;-----------------
;定义功能常量，以下 3 行注释其中 2 行，使一行有效
```

```
MASTER_SLAVE   EQU 0            ;汇编后的程序代码下载到主单片机中
;MASTER_SLAVE   EQU 1            ;汇编后的程序代码下载到从单片机 #1 中
;MASTER_SLAVE   EQU 2            ;汇编后的程序代码下载到从单片机 #2 中
;-----------------
;定义波特率自动重装数常量
;以下波特率是 PCON.7 = 0 时的数值，若使 PCON.7 = 1 可将波特率加倍
;RELOAD_8BIT_DATA   EQU 0FFH    ;Fosc=22.1184MHz, Baud = 57600
;RELOAD_8BIT_DATA   EQU 0FBH    ;Fosc=18.432MHz, Baud=9600, 1T 运行时 Baud=115200
RELOAD_8BIT_DATA    EQU 0F6H    ;Fosc=18.432MHz, Baud=4800, 1T 运行时 Baud=57600
;RELOAD_8BIT_DATA   EQU 0FFH    ;Fosc=11.059MHz, Baud = 28800、
;----------------------------------------------------------------------------
;定义特殊功能寄存器
AUXR     EQU 8EH
;AUXR 特殊功能寄存器的 bit3 是 SPI 中断允许控制位 ESPI
;IE 特殊功能寄存器的 bit5 是ADC和SPI两个中断共享的总中断允许控制位 EADC_SPI
;要产生 SPI 中断，需要 ESPI/EADC_SPI/EA 都为 1
;----------------------------------------------------------------------------
;定义 SPI 特殊功能寄存器，详细说明见本程序的后部
SPCTL           EQU 85H
SPSTAT          EQU 84H
SPDAT           EQU 86H

EADC_SPI        EQU IE.5
;----------------------------------------------------------------------------
;定义 SPI 脚
SCLK            EQU P1.7
MISO            EQU P1.6
MOSI            EQU P1.5
SS              EQU P1.4

Slave1_SS       EQU P1.2
Slave2_SS       EQU P1.3
;----------------------------------------------------------------------------
;定义单片机管脚
LED_MCU_START   EQU P3.4
;----------------------------------------------------------------------------
;定义变量
Flags           EQU 20H
SPI_Receive     EQU Flags.0          ;SPI 端口收到数据标志位

T0_10mS_count   EQU 30H              ;该变量用于保存 10 毫秒计数(T0 中断次数)
SPI_buffer      EQU 31H              ;该变量用于保存 SPI 端口收到的数据
;----------------------------------------------------------------------------
    ORG    0000H
    AJMP   MAIN
;----------------------------------------------------------------------------
```

```
    ORG    000BH                    ;定时器0 中断服务程序入口
    AJMP   timer0_Routine
;-----------------------------------------------------------------------------
    ORG    002BH                    ;ADC_SPI 中断服务程序入口
    AJMP   ADC_SPI_Interrupt_Routine
;-----------------------------------------------------------------------------
    ORG    0080H
MAIN:
    CLR    LED_MCU_START            ;点亮 MCU 开始工作指示灯
    MOV    SP, #7FH
    ACALL Initial_System            ;系统初始化
if MASTER_SLAVE == 0
        CLR    Slave1_SS            ;选择从单片机 #1 为当前的从单片机
Check_RS232:
        JNB    RI, Master_Check_SPI ;判 RS-232 串口中收到数据否
        ;主单片机 RS-232 串口已收到新的数据
        ACALL Get_Byte_From_RS232   ;主单片机将RS-232串口中收到的数据送到累加器 A
;         ACALL RS232_Send_Byte     ;调试用，将累加器 A 中的数据发送到 PC 机
;         SJMP  Check_RS232         ;调试用
        ACALL SPI_Send_Byte         ;主单片机将累加器 A 中的数据发送到从机 SPI
        SJMP  Check_RS232
Master_Check_SPI:
        JNB    SPI_Receive, Check_RS232 ;判收到从 SPI 发回的数据否
        ;主单片机 SPI 端口已收到新的数据
        MOV    A, SPI_buffer        ;将"从 SPI 发回的数据"送到累加器 A
        CLR    SPI_Receive          ;清0 主单片机 SPI 端口收到数据标志位
        ACALL RS232_Send_Byte       ;将累加器 A 中的数据发送到 PC 机
        SJMP  Check_RS232
else
Slave_Check_SPI:
        JNB    SPI_Receive, Slave_Check_SPI ;判收到主 SPI 发回的数据否
        ;从单片机 SPI 端口已收到新的数据
        MOV    A, SPI_buffer        ;取"主单片机 SPI 端口发的数据"
        CLR    SPI_Receive          ;清0 从单片机 SPI 端口收到数据标志位
    if MASTER_SLAVE == 2
        ADD   A, #1                 ;如果是从单片机 #2，就把收到的数据加1
    endif
        MOV    SPDAT, A             ;将收到数据送 SPDAT，准备下一次通讯时发回
        SJMP  Slave_Check_SPI
endif
;-----------------------------------------------------------------------------
if MASTER_SLAVE == 0
timer0_Routine:
        PUSH  PSW                   ;保存断点现场
        PUSH  ACC
```

```
        MOV   TH0, #0C4H              ;重装数 = 65536-15360 = 50176 = C400H
                                      ;晶振频率=18.432MHz时， 每 10mS 中断 1 次
        INC   T0_10mS_count           ;10 毫秒计数(T0 中断次数) + 1
        MOV   A, #0C7H                ;0C8H = 199，检测是否中断了 200 次 (2秒)
        CLR   C
        SUBB  A, T0_10mS_count
        JNC   timer0_Exit
        CPL   Slave1_SS               ;改变当前选择的从单片机
        CPL   Slave2_SS
        MOV   T0_10mS_count, #0       ;清0 10 毫秒计数(T0 中断次数)

timer0_Exit:
        POP   ACC                     ;恢复断点现场
        POP   PSW
        RETI
else
timer0_Routine:                       ;本程序中从单片机不需要使用定时器0
        RETI
endif

;-----------------------------------------------------------------------------
ADC_SPI_Interrupt_Routine:            ;ADC_SPI 中断服务程序
    ;SPI 中断服务程序
    MOV   SPSTAT, #11000000B          ;0C0H，清0 标志位 SPIF 和 WCOL
                  ;特别注意：是向标志位 SPIF/WCOL 写1，将 SPIF/WCOL 清成0
                  ;特别注意：不是向标志位 SPIF/WCOL 写0，将 SPIF/WCOL 清成0
    MOV   A, SPDAT                    ;保存收到的数据
    MOV   SPI_buffer, A
    SETB  SPI_Receive                 ;树立 SPI 端口收到数据标志
    RETI
;-----------------------------------------------------------------------------
Initial_System:
    ACALL Initial_UART                ;初始化串口
    ACALL Initial_SPI                 ;初始化 SPI

    SETB  TR0                         ;启动 T0
    SETB  ET0                         ;开 T0 中断

    MOV   Flags, #0                   ;清标志字
    SETB  EA                          ;开总中断
    RET
;-----------------------------------------------------------------------------
Initial_UART:                         ;初始化串口
; SCON  Bit:   7       6     5     4     3     2     1    0
;            SM0/FE   SM1   SM2   REN   TB8   RB8   Tl   Rl
    MOV   SCON, #50H                  ;0101,0000 8位可变波特率，无奇偶校验
```

```
     MOV   TMOD, #21H                  ;T1 为自动重装模式
     MOV   TH1, #RELOAD_8BIT_DATA
     MOV   TL1, #RELOAD_8BIT_DATA
;    MOV   PCON, #80H                  ;取消本行指令注释，波特率加倍。

     ;使以下两行有效，波特率快 12 倍，即波特率 = 4800*12=57600
     MOV   A, #01000000B               ;T1 以 1T 的速度计数，是普通 8051 的 12 倍
     ORL   AUXR, A

     SETB  TR1                         ;启动定时器 1 开始计数
     RET
;-----------------------------------------------------------------------------
Initial_SPI:                          ;初始化 SPI
if MASTER_SLAVE == 0
    MOV   SPCTL,#11111100B            ;0FCH，忽略 SS 脚，设为主机
    ;SSIG=1: 忽略 SS 脚
    ;SPEN=1：允许 SPI 工作
    ;DORD=1：先传低位 LSB
    ;MSTR=1：设为主机

    ;CPOL=1：SPI 空闲时 SPICLK = 1，前跳变沿是下降沿，后跳变沿是上升沿。
    ;CPHA=1：数据由 SPICLK 前跳变沿驱动到 SPI 口线，SPI 模块在后跳变沿采样数据。
    ;SPR1，SPR0 = 00：主模式时 SPI 时钟源选择为 fosc/4
else
    MOV   SPCTL,#01101100B            ;6CH，设为从机，由 SS 脚决定是否已被选中
    ;SSIG=0: 由 SS 脚决定主模式或从模式。
    ;SPEN=1：允许 SPI 工作
    ;DORD=1：先传低位 LSB
    ;MSTR=0：设为从机

    ;CPOL=1：SPI 空闲时 SPICLK = 1，前跳变沿是下降沿，后跳变沿是上升沿。
    ;CPHA=1：数据由 SPICLK 前跳变沿驱动到 SPI 口线，SPI 模块在后跳变沿采样数据。
    ;SPR1，SPR0 = 00：主模式时 SPI 时钟源选择为 fosc/4
endif
    MOV   SPSTAT,#11000000B           ;清 0 标志位 SPIF(SPSTAT.7)，WCOL(SPSTAT.6)
                                      ;向该两个标志位写"1" 会将它们清 0
    MOV   A, #00001000B
    ORL   AUXR, A                     ;令 ESPI(AUXR.3)=1,允许 SPIF(SPSTAT.7)产生中断
    SETB  EADC_SPI                    ;开 ADC 中断和 SPI 中断共享的总中断控制位
    RET
;-----------------------------------------------------------------------------
RS232_Send_Byte:                      ;RS232 串口发送一个字节
    CLR   TI                          ;清零串口发送中断标志
    MOV   SBUF, A
RS232_Send_Wait:
    JNB   TI, RS232_Send_Wait         ;等待发送完毕，未发送完毕跳回本行
```

```
    CLR    TI                          ;清零串口发送中断标志
    RET
;-----------------------------------------------------------------------------
;此段程序只有主 MCU 调用
SPI_Send_Byte:                         ;SPI 发送一个字节
    CLR    EADC_SPI                    ;关 ADC 中断和 SPI 中断共享的总中断控制位
    MOV    SPDAT, A                    ;SPI 发送数据
SPI_Send_Byte_Wait:
    MOV    A, SPSTAT                   ;等待 SPIF=1 即等待 SPI 发送完毕
    ANL    A, #80H
    JZ     SPI_Send_Byte_Wait
    SETB   EADC_SPI                    ;开 ADC 中断和 SPI 中断共享的总中断控制位
    RET
;-----------------------------------------------------------------------------
Get_Byte_From_RS232:                   ;取 RS-232 串口中收到的数据累加器 A
    MOV    A, SBUF
    CLR    RI
    RET
;-----------------------------------------------------------------------------
    END
;-----------------------------------------------------------------------------
;更详细的资料可以参阅 STC12C5410AD.pdf (中文使用说明)。
;
;SPI 控制寄存器
;          7       6       5       4       3       2       1       0
;SPCTL    SSIG    SPEN    DORD    MSTR    CPOL    CPHA    SPR1    SPR0
;
;SSIG: 忽略SS脚，如果 SSIG=1，由 MSTR 位决定主模式或从模式，
;        如果 SSIG=0，由 SS 脚决定主模式或从模式。
 ;SPEN：SPI使能位。如果 SPEN=0，SPI功能被禁止，SPI 脚用作普通IO口
;DORD：SPI 数据传输顺序。
;        1：先传低位 LSB
;        0：先传高位 MSB
;MSTR：主/从模式选择位
;CPOL：SPI 时钟信号极性选择位
;         1：SPI 空闲时 SPICLK = 1，前跳变沿是下降沿，后跳变沿是上升沿。
;         0：SPI 空闲时 SPICLK = 0，前跳变沿是上升沿，后跳变沿是下降沿。
;CPHA：SPI 时钟信号相位选择位
;         1：数据由 SPICLK 前跳变沿驱动到口线，后跳变沿采样。
;        0：当 SS 脚为低(SSIG=0)时数据被驱动到口线，并且在 SPICLK 后跳变沿数据
;             被改变(被驱动到口线)，在 SPICLK 前跳变沿数据被采样。注意：SSIG = 1
;            时操作未定义。
;SPR1-SPR0：主模式时 SPI 时钟速率选择
;        00：fosc/4
;        01：fosc/16
;        10：fosc/64
```

```
;        11：fosc/128
;
;      当 CPHA=0，SSIG 必须等于零并且在传输时 SS 脚也必须一直保持为低。当 SS 有效
;(=0)时向 SPDATA 寄存器写数据就会发生写冲突错误，WCOL 标志被置 1。
;      当 CPHA=1，SSIG 可以等于 0 或 1。如果 SSIG=0，SS 脚在连续的传输时为 0(可以
;一直保持为 0)。当系统中只有一个主和一个从 SPI 时，这是首选配置。
;-------------------
;SPI 状态寄存器
;           7      6      5      4      3      2      1      0
;SPSTAT   SPIF   WCOL     -      -      -      -      -      -
;SPIF：SPI 传输结束标志。当一次传输结束时， SPIF 被置 1，如果 SPI 中断被打开：
;        ESPI(AUXR.3)=1，EADC_SPI(IE.5)=1，EA(IE.7)=1， 就引起中断。如果原来 SPI
;        由 SS 脚确定为是主模式(SSIG=0,SS=1)，当 SS 变成 0 时，SPIF 也会被置 1，
;        表示"模式改变"。向 SPIF 位写 1 将该标志清 0。
;WCOL：SPI 写冲突标志。当一个数据还在传输时，又向数据寄存器 SPDAT 写入数据，WCOL
;        就会被被置 1。向 WCOL 位写 1 将该标志清 0。
;-------------------
;SPI 主/从模式选择
;
;SPEN SSIG SS MATR    模式    MISO  MOSI SPICLK     注释
;  0    X    X    X   禁止SPI   输入   输入   输入      禁止SPI
;  1    0    0    0     从      输出   输入   输入      被选为从
;  1    0    1    0    未选从   输入   输入   输入      从，但没有被选中
;  1    0    0  1->0    从      输出   输入   输入      由主变为从
;  1    0    1    1     主      输入   输出   输出
;  1    1    X    0     从      输出   输入   输入      从
;  1    1    X    1     主      输入   输出   输出      主
```

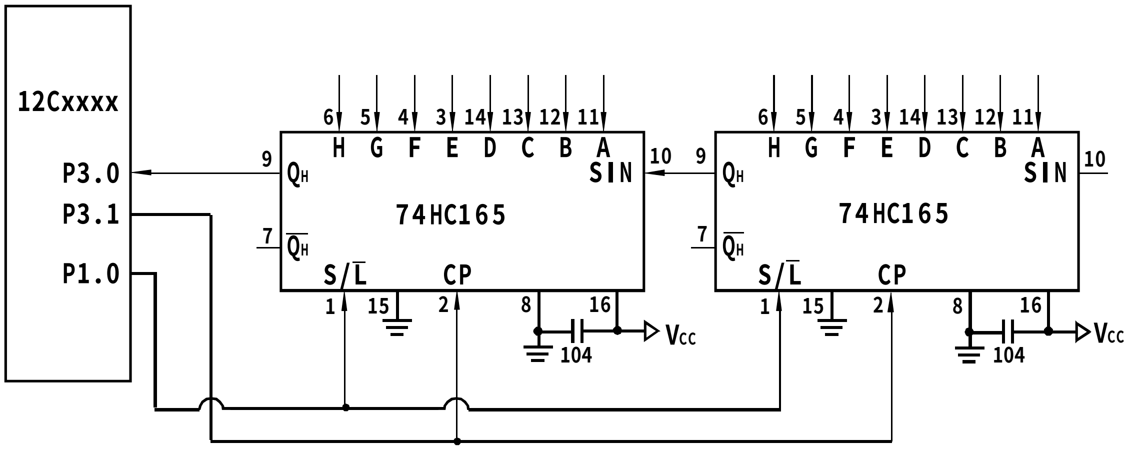
用串行口扩展 I/O 接口

STC12Cxxxx 串行口的方式 0 可用于 I/O 扩展。如果在应用系统中，串行口未被占用，那么将它用来扩展并行 I/O 口是一种经济、实用的方法。

在操作方式 0 时，串行口作同步移位寄存器，其波特率是固定的，为 fosc/12（fosc 为振荡器频率)。数据由 RXD 端（P3.0）出入，同步移位时钟由 TXD 端（P3.1）输出。发送、接收的是 8 位数据，低位在先。

一、用 74HC165 扩展并行输入口

下图是利用两片 74LS165 扩展二个 8 位并行输入口的接口电路图。



74HC165 是 8 位并行置入移位寄存器。当移位 / 置入端(S/$\overline{L}$)由高到低跳变时，并行输入端的数据置入寄存器；当 S/$\overline{L}$=1，且时钟禁止端（第 15 脚）为低电平时，允许时钟输入，这时在时钟脉冲的作用下，数据将由 Q_A 到 Q_H 方向移位。

上图中，TxD(P3.1)作为移位脉冲输出端与所有 74LS165 的移位脉冲输入端 CP 相连；RxD（P3.0）作为串行输入端与 74HC165 的串行输出端 Q_H 相连；P1.0 用来控制 74HC165 的移位与置入而同 S/$\overline{L}$ 相连；74HC165 的时钟禁止端（15 脚）接地，表示允许时钟输入。当扩展多个 8 位输入口时，两芯片的首尾（Q_H 与 S_{IN}）相连。

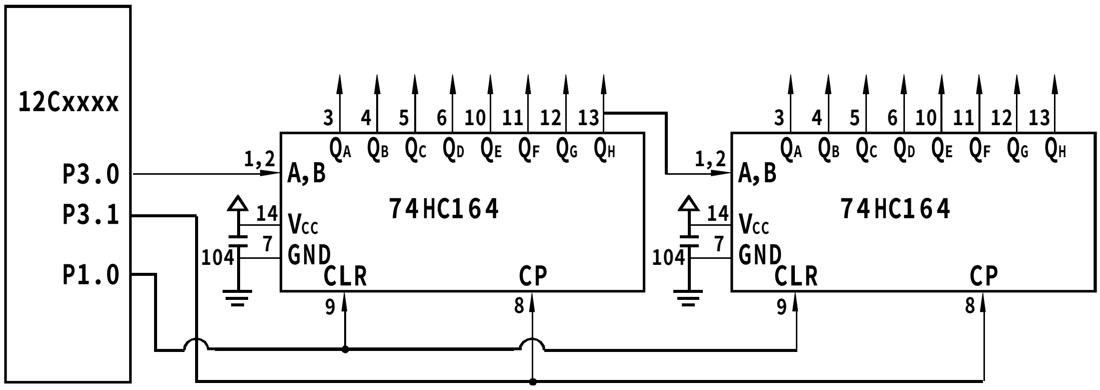
下面的程序是从16位扩展口读入5组数据(每组二个字节)，并把它们转存到内部RAM 20H开始的单元中。

```
         MOV   R7，#05H             ；设置读入组数
         MOV   R0，#20H             ；设置内部 RAM 数据区首址
START：  CLR   P1.0                 ；并行置入数据，S/L=0
         SETB  P1.0                 ；允许串行移位 S/L=1
         MOV   R1，#02H             ；设置每组字节数，即外扩 74HC165 的个数
RXDATA： MOV   SCON，#00010000B     ；设串行方式 0，允许接收，启动接收过程
WAIT：   JNB   RI，WAIT             ；未接收完一帧，循环等待
         CLR   RI                   ；清 RI 标志，准备下次接收
         MOV   A，SBUF              ；读入数据
         MOV   @R0，A               ；送至 RAM 缓冲区
         INC   R0                   ；指向下一个地址
         DJNZ  R1，RXDATA           ；为读完一组数据，继续
         DJNZ  R7，START            ；5 组数据未读完重新并行置入
         ……                         ；对数据进行处理
```

上面的程序对串行接收过程采用的是查询等待的控制方式，如有必要，也可改用中断方式。从理论上讲，按上图方法扩展的输入口几乎是无限的，但扩展的越多，口的操作速度也就越慢。

二、用 74HC164 扩展并行输出口

74LS164 是 8 位串入并出移位寄存器。下图是利用 74LS164 扩展二个 8 位输出口的接口电路。

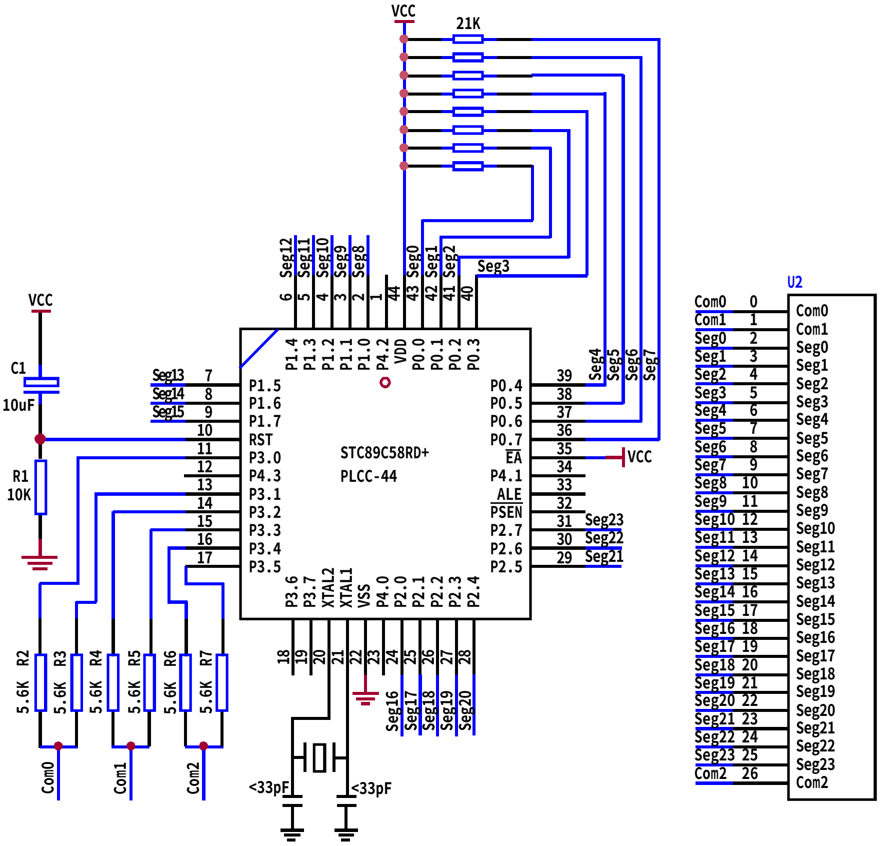


当单片机串行口工作在方式 0 的发送状态时，串行数据由 P3.0（RxD）送出，移位时钟由 P3.1（TxD）送出。在移位时钟的作用下，串行口发送缓冲器的数据一位一位地移入74HC164中。需要指出的是，由于74HC164无并行输出控制端，因而在串行输入过程中，其输出端的状态会不断变化，故在某些应用场合，在74HC164的输出端应加接输出三态门控制，以便保证串行输入结束后再输出数据。

下面是将 RAM 缓冲区 30H、31H 的内容串行口由 74HC164 并行输出的子程序。

```
START：  MOV    R7，#02H      ；设置要发送的字节个数
         MOV    R0，#30H      ；设置地址指针
         MOV    SCON，#00H    ；设置串行口方式0
SEND：   MOV    A，@R0
         MOV    SBUF，A       ；启动串行口发送过程
WAIT：   JNB    TI，WAIT      ；一帧数据未发送完，循环等待
         CLR    TI
         INC    R0            ；取下一个数
         DJNZ   R7，SEND
         RET
```

STC 单片机普通 I/O 口驱动 LCD 显示



本资料不提供技术支持，请自行消化吸收

```
NAME LcdDriver
$include(STC89C51RC.h)

;*****************************************************************************************
;the LCD is 1/3 duty and 1/3 bias; 3Com*24Seg; 9 display RAM;
;
;                    Bit7    Bit6    Bit5    Bit4    Bit3    Bit2    Bit1    Bit0
;Com0:  Com0Data0:   Seg7    Seg6    Seg5    Seg4    Seg3    Seg2    Seg1    Seg0
;       Com0Data1:   Seg15   Seg14   Seg13   Seg12   Seg11   Seg10   Seg9    Seg8
;       Com0Data2:   Seg23   Seg22   Seg21   Seg20   Seg19   Seg18   Seg17   Seg16
;Com1:  Com1Data0:   Seg7    Seg6    Seg5    Seg4    Seg3    Seg2    Seg1    Seg0
;       Com1Data1:   Seg15   Seg14   Seg13   Seg12   Seg11   Seg10   Seg9    Seg8
;       Com1Data2:   Seg23   Seg22   Seg21   Seg20   Seg19   Seg18   Seg17   Seg16
;Com2:  Com2Data0:   Seg7    Seg6    Seg5    Seg4    Seg3    Seg2    Seg1    Seg0
;       Com2Data1:   Seg15   Seg14   Seg13   Seg12   Seg11   Seg10   Seg9    Seg8
;       Com2Data2:   Seg23   Seg22   Seg21   Seg20   Seg19   Seg18   Seg17   Seg16
;*****************************************************************************************
;Com0:  P3^0,P3^1   when P3^0 = P3^1 = 1       then Com0=VCC(=5V);
;                        P3^0 = P3^1 = 0       then Com0=GND(=0V);
;                        P3^0 = 1, P3^1=0      then Com0=1/2 VCC;
;Com1:  P3^2,P3^3   the same as the Com0
;Com2:  P3^4,P3^5   the same as the Com0
;

sbit SEG0  =P0^0
sbit SEG1  =P0^1
sbit SEG2  =P0^2
sbit SEG3  =P0^3
sbit SEG4  =P0^4
sbit SEG5  =P0^5
sbit SEG6  =P0^6
sbit SEG7  =P0^7
sbit SEG8  =P1^0
sbit SEG9  =P1^1
sbit SEG10 =P1^2
sbit SEG11 =P1^3
sbit SEG12 =P1^4
sbit SEG13 =P1^5
sbit SEG14 =P1^6
sbit SEG15 =P1^7
sbit SEG16 =P2^0
sbit SEG17 =P2^1
sbit SEG18 =P2^2
sbit SEG19 =P2^3
```

```
sbit SEG20 =P2^4
sbit SEG21 =P2^5
sbit SEG22 =P2^6
sbit SEG23 =P2^7
;***************************************************************************************

;======Interrupt===============================
       CSEG AT 0000H
       LJMP start

       CSEG AT 000BH
       LJMP int_t0

;======register===============================
lcdd_bit SEGMENT BIT
       RSEG lcdd_bit
       OutFlag:       DBIT 1          ;the output display reverse flag

lcdd_data SEGMENT DATA
      RSEG lcdd_data
      Com0Data0:     DS   1
      Com0Data1:     DS   1
      Com0Data2:     DS   1
      Com1Data0:     DS   1
      Com1Data1:     DS   1
      Com1Data2:     DS   1
      Com2Data0:     DS   1
      Com2Data1:     DS   1
      Com2Data2:     DS   1
      TimeS:         DS   1

;======Interrupt Code=========================
t0_int SEGMENT CODE
     RSEG  t0_int
     USING 1
;*****************************************************************
;Time0 interrupt
;ths system crystalloid is 22.1184MHz
;the time to get the Time0 interrupr is 2.5mS
;the whole duty is 2.5mS*6=15mS, including reverse
;*****************************************************************
int_t0:
    ORL  TL0,#00H
    MOV  TH0,#0EEH
    PUSH ACC
    PUSH PSW
```

```
    MOV   PSW,#08H
    ACALL OutData
    POP   PSW
    POP   ACC
    RETI

;======SUB CODE================================
uart_sub SEGMENT CODE
        RSEG  uart_sub
        USING 0
;******************************************************************
;initial the display RAM data
;if want to display other,then you may add other data to this RAM
;Com0:   Com0Data0,Com0Data1,Com0Data2
;Com1:   Com1Data0,Com1Data1,Com1Data2
;Com2:   Com2Data0,Com0Data1,Com0Data2
;*******************************************************************
InitComData:                    ;it will display "11111111"
    MOV Com0Data0,#24H
    MOV Com0Data1,#49H
    MOV Com0Data2,#92H
    MOV Com1Data0,#92H
    MOV Com1Data1,#24H
    MOV Com1Data2,#49H
    MOV Com2Data0,#00H
    MOV Com2Data1,#00H
    MOV Com2Data2,#00H
    RET

;********************************************************************
;reverse the display data
;********************************************************************
RetComData:
    MOV  R0,#Com0Data0              ;get the first data address
    MOV  R7,#9
RetCom_0:
    MOV  A,@R0
    CPL  A
    MOV  @R0,A
    INC  R0
    DJNZ R7,RetCom_0
    RET
```

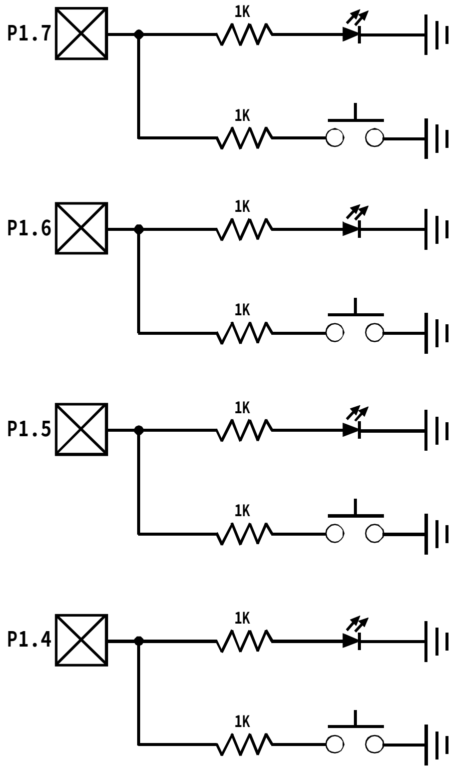
```
;*************************************************************************
;get the display Data and send to Output register
;*************************************************************************
OutData:
    INC  TimeS
    MOV  A,TimeS
    MOV  P3,#11010101B              ;clear display,all Com are 1/2VCC and invalidate
    CJNE A,#01H,OutData_1           ;judge the duty
    MOV  P0,Com0Data0
    MOV  P1,Com0Data1
    MOV  P2,Com0Data2
    JNB  OutFlag,OutData_00
    MOV  P3,#11010111B              ;Com0 is work and is VCC
    RET
OutData_00:
    MOV  P3,#11010100B              ;Com0 is work and is GND
    RET
OutData_1:
    CJNE A,#02H,OutData_2
    MOV  P0,Com1Data0
    MOV  P1,Com1Data1
    MOV  P2,Com1Data2
    JNB  OutFlag,OutData_10
    MOV  P3,#11011101B              ;Com1 is work and is VCC
    RET
OutData_10:
    MOV  P3,#11010001B              ;Com1 is work and is GND
    RET
OutData_2:
    MOV  P0,Com2Data0
    MOV  P1,Com2Data1
    MOV  P2,Com2Data2
    JNB  OutFlag,OutData_20
    MOV  P3,#11110101B              ;Com2 is work and is VCC
    SJMP OutData_21
OutData_20:
    MOV  P3,#11000101B              ;Com2 is work and is GND
OutData_21:
    MOV   TimeS,#00H
    ACALL RetComData
    CPL   OutFlag
    RET
```

```
;======Main Code================================
uart_main SEGMENT CODE
        RSEG  uart_main
        USING 0

start:
           MOV    SP,#40H
           CLR    OutFlag
           MOV    TimeS,#00H
           MOV    TL0,#00H
           MOV    TH0,#0EEH
           MOV    TMOD,#01H
           MOV    IE,#82H
           ACALL  InitComData
           SETB   TR0
Main:
     NOP
     SJMP Main

END
```

一个 I/O 口驱动发光二极管并扫描按键



利用 STC12C5410 系列单片机的 I/O 口可设置成弱上拉,强上拉(推挽)输出,仅为输入(高阻),开漏四种模式的特性,可以利用 STC12C5410 系列单片机的 I/O 口同时作为发光二极管驱动及按键检测用，可以大幅节省 I/O 口。

当驱动发光二极管时，将该 I/O 口设置成强上拉推挽输出，输出高即可点亮发光二极管

当检测按键时，将该 I/O 口设置成弱上拉，然后输出 1，再读外部口的状态，即可检测按键

附录 K: 指令系统与程序设计

执行软件是微型计算机与通用数字集成电路的主要区别，也是微电子技术区别于通用电器和电子技术的根本特征。

软件是由具有一定意义的指令组成的。一台计算机所执行的指令集合就是它的指令系统。指令系统是计算机厂商定义的，它成为应用计算机必须理解和遵循的标准。每种计算机都有自己专用的指令系统。

指令常以英文名称或缩写形式作为助记符。用助记符表示的指令称为汇编语言，用汇编语言编写的程序称为汇编语言程序。

目前单片机主要使用汇编语言，指令系统的学习和应用是使用单片机的重要前提。

STC89 系列单片机与 MCS-51 系列在软件上完全兼容，编制的汇编语言程序可运行于这两种系列单片机。也就是说，STC89 系列单片机采用的也是 MCS-51 指令系统。本章详细介绍该指令系统及其编程方法。

1 指令格式及其符号说明

指令的表示方法称为指令格式。一条指令通常由两部分组成：操作码和操作数。操作码规定指令执行什么操作，而操作数是操作的对象。操作数可以是一个具体的数据，也可以是存储数据的地址或寄存器。指令的基本格式如下：

操作码	操作数（地址码、寄存器或立即数）

汇编语言编写的程序必须翻译成单片机可执行的机器码。根据机器码的长短，可分为单字节、双字节和 3 字节等不同长度的指令。

1．单字节指令

指令系统中有些指令的功能很专一而明确，不需要具体指定操作数，便形成了单字节指令。单字节指令的机器码只有一个字节，操作码和操作数同在其中。例如，指令 INC DPTR，功能为数据指针加 1，指令码为

1010 0011	A3H

有些指令的操作数在工作寄存器 R0～R7 中，寄存器的编码可用 3 位二进制数表示。例如，指令 MOV A，Rn，功能是工作寄存器向累加器传输数据，指令码为

1110 1rrr

用 rrr 表示工作寄存器的二进制编码。对于不同的工作寄存器，单字节的机器码如下表所列。

指令MOV A，Rn 指令码

指令	指令码（机器码）	
	二进制	十六进制
MOV A，R0	1110 1000	E8H
MOV A，R1	1110 1001	E9H
MOV A，R2	1110 1010	EAH
MOV A，R3	1110 1011	EBH
MOV A，R4	1110 1100	ECH
MOV A，R5	1110 1101	EDH
MOV A，R6	1110 1110	EEH
MOV A，R7	1110 1111	EFH

2．双字节指令

双字节指令的第一字节是操作码，第二字节是操作数。例如，指令MOV A，#data，功能是将立即数传送到A，指令码为

0111 0100
立即数

例如，指令MOV A，#35H 的指令码为7435H。

3．3字节指令

3 字节指令中，操作码占一字节，操作数占两字节。其中操作数既可以是数据，也可以是地址。例如，指令ANL direct，#data，功能是直接地址单元中的内容与立即数进行“与”操作，结果存于直接地址单元，指令码为

0101 0011
直接地址
立即数

例如，指令ANL 35H，#20H 的机器码为533520H。

在介绍指令之前，先将指令中使用的一些符号意义作简要说明。

- Rn ——当前工作寄存器R0～R7，即n＝0～7，在指令中表示寄存器寻址方式。
- Ri ——间接寻址的寄存器R0 和R1，即i=0,1，在指令中表示间接寻址方式。
- direct ——8 位直接地址，在指令中表示直接寻址方式，寻址范围为00H～FFH。
- #data ——8 位立即数，表示立即数寻址方式。
- #data16 ——16 位立即数，表示立即数寻址方式。
- addr16 ——16 位目的地址，只限用于LCALL 和LJMP 指令。
- addr11 ——11 位目的地址，只限用于ACALL 和AJMP 指令。
- rel ——相对转移指令中的偏移量，为8 位带符号补码数，在指令中表示相对寻址方式。
- DPTR ——数据指针，16 位。
- bit ——内部数据RAM 和特殊功能寄存器中的可寻址位。

● A ——表示累加器。
●ACC ——直接寻址方式的累加器。
● B ——寄存器 B。
● C ——进位标志位，可作为位处理器的位累加器，也称为累加位。在指令中代表 CY。
● @ ——间址寄存器的前缀标志。
● / ——加在位地址前面，表示该位状态取反。
● （X）——某个寄存器或某地址单元中的内容。
● （(X)）——由 X 间接寻址单元中的内容。
● ←——箭头右边的内容传送到箭头左边的存储器单元或寄存器中，即表示数据的传送方向。
● →——箭头左边的内容传送到箭头右边的存储器单元或寄存器中，即表示数据的传送方向。

2 寻址方式

指令执行是都要应用操作数。指令必须指明如何取得操作数，也必须指明程序转移目的地址。所谓寻址，就是如何指定操作数所在的单元，或者如何指定程序转移的目的地址。根据指定的方法不同，形成了不同的寻址方式。MCS-51 指令系统有 7 种不同的寻址方式，下面分别介绍。

1． 寄存器寻址方式

寄存器寻址时，指令中操作数为某一寄存器的内容。指定了寄存器，就指定了操作数。该寻址方式中，用符号名称表示寄存器。

寄存器寻址方式所使用的寄存器包括：

1） 工作寄存器 R0～R7，只能寻址当前寄存器组，即由 PSW 中的 RS1 和 RS0 位的状态对应的当前寄存器组。

2） 部分特殊功能寄存器，例如 A、AB 寄存器对以及数据指针 DPTR 等。

例如：

INC R0；(R0) ←（R0）+1

其功能是把寄存器 R0 的容量加 1，再送回 R0 中。由于操作数在 R0 中，指定了 R0，也就得到了操作数。

2． 直接寻址方式

直接寻址时，指令中操作数部分直接给出了操作数地址。例如：

MOV A，4AH；(A) ←（4AH）

该指令的功能是把片内 RAM 4AH 单元的内容送入累加器 A。指定了地址 4AH，也就的到了操作数。

直接寻址中的操作数以存储单元形式出现，因此直接寻址方式只能用 8 位二进制数表示的地址，寻址范围只限于内部 RAM，即：

1） 片内 RAM 低 128 单元，在指令中直接以单元地址形式给出。

2）特殊功能寄存器。特殊功能寄存器除了用单元地址形式给出外，还可以用寄存器的名称符号表示。应当指出，直接寻址方式是访问特殊功能寄存器的主要方法。例如：

MOV A，P1 ；(A) ←（P1）
MOV A，90H ；(A) ←（90H）

由于 SFR P1 的地址为 90H，两条指令本质上是一样的，有相同的机器码，都是直接寻址方式。

3． 寄存器间接寻址方式

寄存器间接寻址时，指令中给出的寄存器内容为操作数地址，而不是操作数本身，即寄存器

为地址指针。

为区别寄存器寻址和寄存器间接寻址，在寄存器间接寻址中，应在寄存器的名称前加前缀@。例如：

```
MOV  R1，#60H
MOV  A，@R1
```

该指令的功能是将60H单元的内容送入累加器A。

2) 外部数据RAM空间的256个单元。例如：

```
MOVX  A，@R1
```

由R1中内容指定的外RAM单元内容送入累加器A。

用DPTR作间址寄存器，其形式为@DPTR，可寻址外部RAM 64 KB（0000H～FFFH）。例如：

```
MOVX  @DPTR，A
```

将累加器的内容传送到由DPTR内容指定的片外RAM16位地址单元。

堆栈操作指令（PUSH和POP）也应算作是寄存器间接寻址，即以堆栈指针（SP）作间址寄存器的间接寻址方式，只不过SP不出现在堆栈操作指令中。

4. 立即寻址方式

立即寻址方式是由指令直接给定操作数的方式。例如：

```
MOV  A，#48H；(A) ← #48H
```

其中#作为立即数的标志符。指令的功能是将数据48H送入累加器A。

除8位立即数外，MCS-51指令系统中还有一条16位立即数传送指令，以#data16表示16位立即数。该指令为

```
MOV  DPTR，#data16
```

其功能是将16位立即数送入数据指针DPTR。例如：

```
MOV  DPTR，#1234H
```

其功能是将12H送入DPH，34H送入DPL。

5. 变址寻址方式

变址寻址方式是以程序计数器PC或数据指针DPTR作为基址寄存器，以累加器A作为变址寄存器，这两者内容之和为有效地址。例如：假定指令执行前（A）= 54H，(DPTR）= 3F21H，执行指令

```
MOVC  A，@A+DPTR
```

其功能是将程序存储器3F75H单元的内容读入累加器A。

这类寻址方式特别适用于查表。DPTR可指向64KB存储空间；@A+PC指向以PC当前值为起始地址的256个字节单元。

对变址寻址方式说明如下：

1) 变址寻址方式只能对程序存储器寻址，或者说它是专门针对程序存储器的寻址方式。

2) 变址寻址指令只有3条，即

```
MOVC  A，@A+DPTR
MOVC  A，@A+PC
JMP       @A+DPTR
```

前两条是程序存储器指令，最后一条是无条件转移指令。

3) 变址寻址方式中的A、DPTR以及PC中的内容为无符号数。

4) 尽管变址寻址方式比较复杂，但变址寻址的指令却都是单字节指令。

6．位寻址方式

位寻址时，操作数是二进制数表示的地址，其位地址出现在指令中。例如：

CLR bit

该指令使地址为 bit 的位单元清 0。

位寻址的寻址范围如下：

1） 片内 RAM 中的位寻址区。 其单元地址为 20H～2FH，共 16 个单元 128 位，位地址为 00H～7FH。对这 128 位的寻址可使用直接地址表示。

2） 特殊功能寄存器的可寻址位。对这些寻址位在指令中常用以下几种表示方法：

- 直接使用位地址，例如 PSW 中的位 5 地址为 D5H；
- 位名称表示法，例如 PSW 的位 5 是 F0 标志位，可使用 F0 表示；
- 特殊功能寄存器符号名称加位数的表示方法，例如 PSW 的位 5 可表示成 PSW.5。

7．相对寻址方式

前面介绍的 6 种寻址方式主要解决操作数的给出问题，而相对寻址方式是为解决程序转移而专门设置的，为转移指令所采用。

相对寻址是以 PC 的相对值为基地址，加上指令中所给定的偏移量，形成有效转换地址。偏移量是带符号的 8 位二进制数，以补码的形式出现。因此，程序的转移范围为 +127～-128。转移目的地址可用如下公式表示：

目的地址 = 转移指令所在地址 + 转移指令字节数 +rel

例如：

SJMP rel；(PC) ← (PC) + 2 + rel

执行这条指令时，程序转移到指令 PC 值加 2 再加 rel 的方向地址处。其中，2 为该指令的字节长度，rel 以 8 位带符号的补码形式出现。

3 指令分类介绍

MCS-51 指令系统共有 111 条指令，分为 5 大类：

- 数据传送类指令（29 条）；
- 算术运算类指令（24 条）；
- 逻辑运算及移位类指令（24 条）；
- 控制转移类指令（17 条）；
- 位操作类指令（17 条）。

1 数据传送类指令

数据传送操作属于复制性质,而不是搬家性质。一般传送类指令的助记符号为 MOV,通用格式为

MOV <目的操作数>,<源操作数>

传送指令中有从右向左传送数据的约定,即指令的右边操作数为源操作数,表达的是数据的来源,而左边的操作数为目的操作数,表达的是传送数据的目的地址。

源操作数可以是：累加器 A、工作寄存器 Rn、直接地址 direct、间址寄存器和立即数。目的操作数可以是：累加器 A、工作寄存器 Rn、直接地址 direct 和间址寄存器。两者只差一个立即数。

在数据传送操作中，除了奇偶标志 P 外，一般不影响程序状态字 PSW（指令直接访问 PSW 除外）。

1．一般传送指令

（1） 以累加器 A 为目的操作数的传送指令

```
MOV   A，Rn           ；(A ) ← (R n )
MOV   A，direct       ；(A) ← (direct)
MOV   A，@Ri          ；(A) ←((Ri))
MOV   A，#vdata       ；(A) ←#data
```

（2） 以工作寄存器为目的操作数的传送指令

```
MOV   Rn，A            ；(R n ) ← (A )
MOV   Rn，direct       ；(Rn) ← (direct)
MOV   Rn，#data        ；(A) ←#data
```

(3） 以直接地址为目的操作数的传送指令

```
MOV   direct，A        ；(direct) ← (A)
MOV   direct，Rn       ；(direct) ← (Rn)
MOV   direct，@Ri；(direct) ← (Ri)
MOV   direct1，direct2；(direct1) ← (direct2)
MOV   direct，#data；  (direct) ←#data
```

(4） 以寄存器间接地址为目的操作数的传送指令

```
MOV   @Ri，A           ；((R i )) ← (A )
MOV   @Ri，direct；((Ri)) ← (direct)
MOV   @Ri，#data       ；((Ri)) ←#data
```

例 1 把 25H 和 10H 数据分别送到片内 RAM20H 和 25H 单元；把 CAH 送 P1 口；将 P1 口内容送 P2 口；将 RAM20H 单元内容送以 R0 间址的存储单元。

```
MOV   20H，#25H        ；(20H) ←#25H
MOV   25H，#10H        ；(25H) ←#10H
MOV   P1，#0CAH        ；(P1) ←#0CAH
MOV   P2，P1           ；(P2) ←P1
MOV   @R0，20H         ；((R0)) ← (20H)
```

操作数的寻址方式如下表所列。

例 1 操作数寻址方式

指令	目的操作数	源操作数
MOV 20H，#25H	直接寻址	立即寻址
MOV 25H，#10H	直接寻址	立即寻址
MOV P1，#0CAH	直接寻址	立即寻址
MOV P2，P1	直接寻址	直接寻址
MOV @R0，20H	间接寻址	直接寻址

2．16 位地址指针传送指令

```
MOV   DPTR，#data16          ；(DPTR) ←#data16
```

这条指令的功能是将 16 位常数送入数据指针 DPTR。这是 MCS-51 指令系统中惟一一条 16 位数据传送指令。DPTR 由 DPH 和 DPL 组成。该指令将高 8 位立即数送入 DPH，低 8 位立即数送入 DPL。

例如：

MOV DPTR，#1992H ；(DPH）←#19H，(DPL）←#92H

也可写成两条 8 位传送指令：

MOV DPH，#19H

MOV DPL，#92H

3．栈操作指令

栈操作指令有进栈 PUSH 和出栈 POP 两条指令：

PUSH direct；(SP) ← (SP) +1；((SP)) ← (direct)

POP direct；(direct) ← ((SP))；(SP) ← (SP) -1

栈操作指令的操作数有两种寻址方式：SP 间接寻址（隐含在指令中）和直接寻址方式。例如：

PUSH B ；B 为直接寻址方式

PUSH DPH ；DPH 为直接寻址方式

对于工作寄存器的栈操作，只能使用 Rn 的当前直接地址，而不能用 Rn 名称，因为栈操作指令不能区别 Rn 的当前组别。如果 Rn 工作在组 1 时，R1 的直接地址为 09H，对 R1 的直接地址为 09H，对 R1 的栈操作应写成：PUSH 09H 或 POP 09H。

4．累加器 A 数据交换指令

(1) 字节交换指令

XCH A，Rn ；(A）⇆ (Rn)

XCH A，direct ；(A) ⇆ (direct)

XCH A，@Ri ；(A）⇆ ((@Ri))

该指令的功能是将 A 与源操作数内容互相交换。

例 2 设 (A) = 92H，(R0) = 20H，(20H) = 12H，执行指令 XCH A，@R0 后，则 (A) =12H，(20H) = 92H。

(2) 半字节交换指令

XCHD A，@Ri ；$(A)_{0\sim3}$ ⇆ $((Ri))_{0\sim3}$

这条指令的功能是将 A 中的低 4 位与 Ri 间址单元内容的低 4 位交换，各自的高 4 位不变。

例 3 设 A 中的内容为 58H，(R0) = 20H，片内 RAM 20H 单元的内容为 65H，执行 XCHD A，@R0 后，则 A 的内容为 55H，片内 RAM 20H 单元内容为 68H。

(3) 累加器 A 高 4 位与低 4 位相互交换指令

SWAP A ；$(A)_{0\sim3}$ ⇆ $(A)_{4\sim7}$

例如，设 A 中的内容为 ABH，执行上述指令后，A 中的内容就变为 BAH。

5．累加器 A 与外部 RAM 传送指令

MOVX A， @Ri ；(A) ← ((Ri))

MOVX A， @DPTR ；(A) ← ((DPTR))

MOVX @Ri，A ；((Ri)) ← (A)

MOVX @DPTR，A ；((DPTR)) ← (A)

单片机与外部 RAM 进行数据交换时，只能通过累加器 A。采用 R0 和 R1 作间址寄存器时，在给定 P2 情况下，可寻址外 RAM 的 256 个单元；采用 DPTR 作间址寄存器时，可寻址外 RAM 的 64KB 空间。

6．累加器 A 与程序存储器传送指令

MOVC A， @A+DPTR；(A) ← ((A) + (D P T R))
MOVC A， @A+PC ；(A) ← ((A) + (P C))

上述两条适龄以DPTR或PC作为基址寄存器，A中的内容为8位无符号数（A称为变址寄存器），将基址寄存器内容与A 中的内容相加，得到一个16 位地址，将该地址指出的程序存储器单元的内容送入累加器A。

例4 程序存储器中有一字形表的首地址为0198H，若要调用表中第一字符，则可用下列指令 ：

```
MOV   DPTR， #0198H    ；设置地址指针
MOV   A， #00H         ；设置变形首址
MOVC  A， @A+DPTR；寻找字形码
MOVX  @R0， A          ；字形码送外字形口
```

例5 根据累加器A的内容（0～3）找出由伪令DB所定义的4个字符中的一个。

```
START：INC A           ；(A) ← (A) +1，单字节指令
       MOVC A,@A+PC    ；(PC) ← (PC) +1，(A) ← ((A) + (PC))，单字节指令
       RET             ；单字节指令
       DB  29H
       DB  0A2H
       DB  92H
       DB  45H
```

DB是伪指令，功能是将右边的单字节数据存入其左边标号地址单元内。如果DB左边没有标号，则DB伪指令的右边字节数据在DB指令的当前地址连续存放。

该子程序在MOVC指令前面有一条INC A指令，其作用是跳过表格中的RET指令。如果指令MOVC所在地址与表格首地址由若干字节隔开，就需要在累加器A中加上相应的数目。本例中A的取值限定在0～3。在调用上述子程序时，若（A）= 02H，则在执行完这段程序后，A中的内容为92H。

PC的当前值是指读取“ MOVC A,@A+PC ”后的PC值，即该指令下面的指令所对应的地址。由于“ MOVC A,@A+PC ”为单字节指令，将该指令所在地址加1，即为PC当前值指向指令RET所在地址。

2 算术运算类指令

MCS-51指令系统具有较强的加、减、乘、除四则运算指令，但只有8位数据运算指令，没有16位数据运算指令。

1. 加法类指令

(1) 加法指令

```
ADD   A，Rn         ；(A ) ← (A ) + (R n )
ADD   A，direct     ；(A) ← (A) + (direct)
ADD   A，@Ri        ；(A) ← (A) + ((Ri))
ADD   A，#data      ；(A) ← (A) +#data
```

上述指令的功能是将累加器A中的内容与源操作数相加，结果存于A中。

当相加结果的第3位和第7位有进位时，分别将AC和CY置1，否则清0。

无符号数相加后，若CY=1，表示溢出；CY=0，表示无溢出。

对于带符号数相加结果的溢出，取决于第7位和第6位。若第7位有进位而第6位没进位，或第7位没进位而第6位有进位，则OV=1；若第7位和第6位都有进位，或都没进位，则OV=0。OV=1表示两个正数相加而和变为负数，或两个负数相加而和变为正数的错误结果。

例如：(A)=0C2H，(R0)=0A9H，执行ADD A，R0 指令，过程表示为

```
        1100 0010
  +)    1010 1001
  ---------------
       10110 1011
```

运算结果（A)=6BH，(AC)=0，(CY)=1，(OV)=1。若0C2H 和0A9H 是两个无符号数，则结果是正确的；若0C2H 和0A9H 是两个带符号的数，由于有溢出，则表明结果是错误的，因为两个负数相加的结果不可能的到正数。

例6　片内RAM 40H和41H单元分别放两个加数，相加结果存放在41H和40H单元。

编制程序如下：

```
      MOV  R0，#40H   ；设置地址指针
      MOV  A, @R0     ；取第一个加数
      INC  R0         ；修改地址指针
      ADD  A, @R0     ；两数相加
      DEC  R0         ；修改地址指针
      MOV  @R0，A     ；存放和的低字节
      INC  R0         ；修改地址指针
      JC   LOOP       ；有进位则转
      MOV  @R0，#00H  ；存放和的高字节
      RET
LOOP：MOV  @R0，#00H  ；存放和的高字节
      RET
```

（2）　带进位的加法指令

```
ADDC A，Rn          ；(A) ← (A) + (Rn) + (CY)
ADDC A，direct      ；(A) ← (A) + (direct) + (CY)
ADDC A，@Ri         ；(A) ← (A) + ((Ri)) + (CY)
ADDC A，#data       ；(A) ← (A) +#data+ (CY)
```

上述4条指令的操作数除了需要加上进位CY外，其余与ADD的4条指令的操作相同。

例7　设A中的内容为C3H，R0的内容为AAH，CY=1，执行指令ADDC A，R0的过程为

```
        1100 0011
        1010 1010
  + )           1
  ---------------
      1 0110 1110
```

结果：A 中的内容为6EH，(AC)=0，(CY)=1，(OV)=1。

（3）　加1指令

```
INC  A          ；(A) ← (A) + 1
INC  Rn         ；(Rn) ← (Rn) + 1
INC  direct     ；(direct) ← (direct) +1
INC  @Ri        ；((Ri)) ← ((Ri)) +1
INC  DPTR       ；(DPTR) ← (DPTR) +1
```

INC指令是把指定的单元内容加1，结果仍存原单元中。加1指令除影响奇偶标志P外，运算结果不影响其他标志位。

加1指令为，当目的操作数是P0～P3口时，数据来自端口锁存器（即为SFR），结果仍写回端

口锁存器。这类以端口为目的操作数的指令被称为“读 - 修改 - 写”指令。

例8 设DPTR的内容为12FEH，执行下列指令：

```
INC  DPTR    ;(DPH)←12H,(DPL)←FFH
INC  DPTR    ;(DPH)←13H,(DPL)←00H
INC  DPTR    ;(DPH)←13H,(DPL)←01H
```

（4） 二 - 十进制调整指令

```
DA   A
```

这是一条专用指令，用于对BCD码十进制加法运算的结果进行修正。MCS-51指令系列系统中没有十进制（BCD）的加法指令，只能借助于二进制加法指令。然而，二进制数的加法用于十进制加法运算时，有时会产生错误结果。例如：

```
1)  6+3=9           2)  8+7=15          3)  8+9=17
      0110                1000                1000
 +)   0011           +)   0111           +)   1001
 ----------          ----------          ----------
      1001                1111              1 0001
```

其中：1）的运算是正确的，因为9的BCD码就是1001；2）的运算结果是不正确的，因为BCD码没有1111；3）的运算结果也是错误的，因为运算结果是11，而不是17。

出错的原因在于，BCD码是4位的二进制编码，而4位二进制编码共有16个编码，但BCD码只用了其中的10个，剩下的6个没有用。通常把这6个没有用的编码（1010，1011，1100，1101，1110，1111）称为无效码。

在BCD码的加法运算中，凡是结果已进入或跳过无效编码区时，其结果都是错误的。相加的结果大于9，说明已进入无效编码区；相加的结果有进位，说明已跳过无效编码区。但不管是哪一种出错情况，相加结果都比正确值小6。出错是由6个无效编码造成的。

为此，对BCD码运算结果进行“加6”调整，才能得到正确的结果。“加6”的条件是：

1） $(A)_{3\sim0}>9$ 或（AC）=1；

2） $(A)_{7\sim4}>9$ 或（CY）=1。

十进制调整指令不影响溢出标志。

例9 设累加器A的内容为1000 1000B（即 BCD码88），工作寄存器R5的内容为1001 1001B（即 BCD码99），（CY）=1。执行下列指令：

```
ADDC A,R5
DA   A
```

第一条加法指令执行后，A中的内容为0010 0010B；（CY）=1，（AC）=1。然后执行十进制调整指令DA A。因为（CY）=1，（AC）=1，所以高4位和低4位均自动加6调整，即

```
          (A) =1000  1000       BCD88
         (R5) =1001  1001       BCD99
   +)                   1
   ------------------------
            1  0010  0010       BCD122
   调整+)       0110  0110       BCD66
   ------------------------
            1  1000  1000       BCD188
```

以上所讲的十进制调整的原理和方法，在具体操作时是通过片内硬件逻辑电路实现的。

例10 设一个加数存于40H和41H单元，另一个加数存于42H和43H单元，和存于40H和41H单元。4位BCD码的加法程序如下（假定相加的结果仍为4位BCD码）：

```
MOV  R0,#40H        ;R0指向加数低字节
```

```
MOV  R1，#42H      ；R1 指向另一个加数低字节
MOV  A，@R0
ADD  A，@R1        ；个位、十位数相加
DA   A             ；十进制调整
MOV  @R0，A        ；存低位和于 40H 单元
INC  R0            ；指针指向百位、千位数
INC  R1
MOV  A，@R0
ADDC A，@R1        ；百位、千位数相加
DA   A
MOV  @R0，A        ；存高位和于 41H 单元
RET
```

2．减法类指令

（1） 带借位减法指令

```
SUBB A，Rn         ；(A) ← (A) — (Rn) — (CY)
SUBB A，direct     ；(A) ← (A) — (direct) — (CY)
SUBB A，@Ri        ；(A) ← (A) — ((Ri)) — (CY)
SUBB A，#data      ；(A) ← (A) —#data — (CY)
```

如果第 7 位借位，则（CY）=1，否则（CY）=0；若第 3 位有借位，则（AC）=1，否则（AC）=0；溢出标志 OV 用于带符号的整数减法，若第 7 位和第 6 位中只有一位有借位，而另一位没有借位，则（OV）=1。（OV）=1 表示一个正数减去一个负数结果为负数，或一个负数减去一个正数为正数的错误结果。当无符号数运算时，溢出标志无意义。

例 11 设累加器 A 中的内容为 0ECH，寄存器 R5 中的内容为 75H，（CY）=1，执行指令 SUBB A，R5，其运算操作过程为

```
        1110 1100
-)      0111 0101
-----------------
        0111 0111
-)             1= (CY)
-----------------
        0111 0110
```

结果：(A) =76H，(CY) =0，(AC) =0 (OV) =1。

（2） 减 1 指令

```
DEC  A         ；(A) ← (A) - 1
DEC  Rn        ；(Rn) ← (Rn) -1
DEC  direct    ；(direct) ← (direct) -1
DEC  @Ri       ；((Ri)) ← ((Ri)) -1
```

减 1 指令的功能是指令单元的内容减 1，结果存于原单元中。除了标志 P 外，本指令不影响其他标志位。

当减 1 指令的目的操作数是 P0～P3 端口时，该指令属于“读 - 修改 - 写”指令，即将端口数据读出，减 1，又送回原端口。

3．乘法和除法指令

（1） 乘法指令

```
MUL  AB        ;(B15~8  A7~0) ← (A) × (A)
```

将A和B的无符号数相乘，16位乘积的低8位存于A，高8位存于B。乘法指令影响3个标志位：(CY)=0；若 (B)=0，则 (OV)=0，若若 (B) ≠ 0，则 (OV)=1；P标志仍按A中的内容设 置。

(2)　除法指令

```
DIV  AB        ;(A) ←商，(B) ←余数
```

将A中的8位无符号数除以B中的8为2无符号数，商存于A，余数存于B。

DIV操作影响3个标志位：(CY)=0；(B)=0（即非法）时 (OV)=1，表明除法没有意义，而其他情况下 (OV)=0；P标志仍取决于A的内容。

3　逻辑运算及移位类指令

MCS-51指令系统能对位和字节操作数进行基本的逻辑运算。下面介绍字节操作数的逻辑运算，有关位操作将在后面介绍。

1．逻辑“与”运算指令

```
ANL  A，Rn           ;(A) ← (A) ∧ (Rn)
ANL  A，direct       ;(A) ← (A) ∧ (direct)
ANL  A，@Ri          ;(A) ← (A) ∧ ((Ri))
ANL  A，#data        ;(A) ← (A) ∧#data
ANL  direct，A       ;(direct) ← (direct) ∧ (A)
ANL  direct，#data   ;(direct) ← (direct) ∧#data
```

例12　已知 (A)=1010 1101B，(R4)=0110 0101B。执行指令ANL A，R4的过程为

```
      (A) =1010 1101
∧) (R4) =0110 0101
------------------
      (A) =0010 0101
```

2．逻辑“或”运算指令

```
ORL  A，Rn           ;(A) ← (A) ∨ (Rn)
ORL  A，direct       ;(A) ← (A) ∨ (direct)
ORL  A，@Ri          ;(A) ← (A) ∨ ((Ri))
ORL  A，#data        ;(A) ← (A) ∨#data
ORL  direct，A       ;(direct) ← (direct) ∨ (A)
ORL  direct，#data   ;(direct) ← (direct) ∨#data
```

例13　将累加器A的高5位送到P1口的高5位，而P1口的低3位保持不变。程序如下：

```
MOV  R2，A           ;暂存A的内容
ANL  A，#0F8H        ;取A的高5位
ANL  P1，#07H        ;取P1的低3位
ORL  P1，A           ;组合P1口内容
MOV  A，R2           ;恢复A的内容
```

3．逻辑“异或”指令

“异或”操作也是按位进行的。当两个操作数相同时，结果为0；不同时，结果为1。运算符号为⊕。

```
XRL  A，Rn           ;(A) ← (A) ⊕ (Rn)
```

XRL　A，direct　　；(A) ← (A) ⊕ (direct)
XRL　A，@Ri　　　；(A) ← (A) ⊕ ((Ri))
XRL　A，#data　　；(A) ← (A) ⊕ #data
XRL　direct，A　　；(direct) ← (direct) ⊕ (A)
XRL　direct，#data　；(direct) ← (direct) ⊕ #data

使用“异或”指令可判别两个数是否相等。若相等，则结果为全0。利用本指令可对目的操作数的某些位取反或保留：用1去“异或”的位，则取反；用0去“异或”的位，则保留。

在MCS-51指令系统中的逻辑“与”、“或”、“异或”运算时，当目的操作数为P0～P3端口时，指令属于“读-修改-写”指令。

4．累加器清0及取反指令

CLR　A　　；(A) ←#00H
CLR　A　　；(A) ← $(\overline{A})$

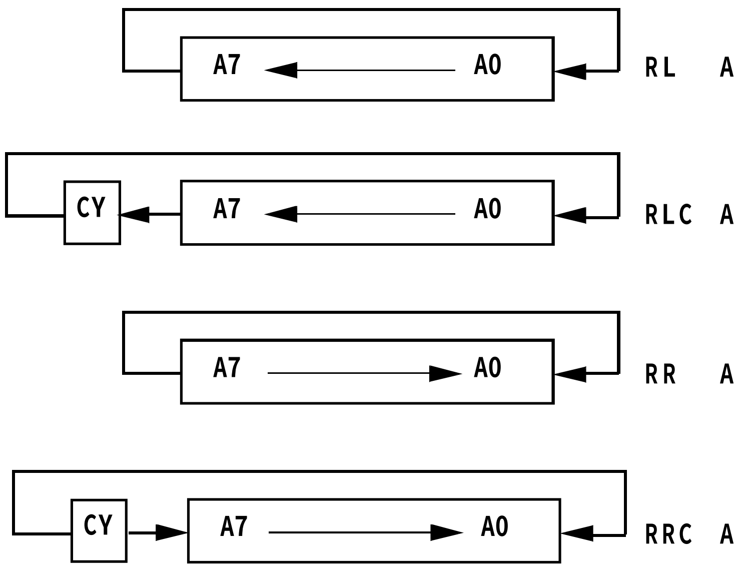
MCS-51指令系统没有“求补”指令，若需要进行“求补”运算，可用“取反加1”运算规则实现。

5．移位指令

MCS-51指令系统的移位操作只对累加器A进行，有左、右小循环和左、右大循环4种：

左小循环　RL　A
右小循环　RR　A
左大循环　RLC　A
右大循环　RRC　A

以上4条指令的操作过程，如下图所示。



移位指令示意图

4　控制转移类类指令

程序的顺序执行是靠PC自动加1实现的。要改变程序的执行顺序，实现分支转向，应通过

强迫改变 P C 值的方法来实现。这就是控制转移类指令的基本功能。

共有两类转移：无条件转移和有条件转移。

1．无条件转移指令

(1)　长转移指令

LJMP addr16　　;（P C）←addr16

这是一条 3 字节指令，指令执行后把 16 位地址（addr16）送入 PC，从而实现了程序的转移。因为转移范围大，可达 64KB，故称为“长转移”。

(2)　绝对转移指令

LJMP addr11　　;（P C）←（P C）+2，$(PC)_{10\sim0}$←addr11

AJMP 指令提供 11 位地址去替换 PC 的低 11 位地址内容，形成新的 PC 值，即转移目的地址。

AJMP 是一条双字节指令，指令的格式为

第一字节	A10	A9	A8	0	0	0	0	1
第二字节	A7	A6	A5	A4	A3	A2	A1	A0

指令提供的 11 位地址中，A7～A0 在第二字节，A10～A8 则占据第一字节的高 3 位，而指令操作码只占第一字节的低 5 位（00001）。AJMP 指令的功能是构造程序转移目的地址，实现程序的转移。其构造新地址的方法是：以指令提供的 11 位地址（A10～A0）去替换 PC 的低 11 位，形成新的 PC 值，即转移目的地址。但要注意，被替换的 PC 值是 AJMP 指令的地址加 2 的 PC 值，即指向 AJMP 下条指令的 PC 值，称为 PC 当前值。例如，在程序存储器的 2070H 单元存放一条绝对转移指令：

2070H　　AJMP　　NEWAD

标号地址 NEWAD 的低 11 位地址为 16AH=001 0110 1010B，构成的指令代码为 216AH，即

0	0	1	0	0	0	0	1
0	1	1	0	1	0	1	0

程序计数器 PC 加 2 的内容为：0010 0000 0111 0010B=2072H，以 11 位绝对地址（16AH）代替 PC 中的低 11 位，形成的转移目的地址为：0010 0001 0110 1010B=216AH。

addr11 是无符号整数，最小值为 000H，最大值为 7FFH，因此绝对转移指令所能转移的最大范围是 2KB。对于“2070H AJMP NEWAD”指令，其转移范围是 2000H～27FFH。

(3)　短转移指令

SJMP rel

SJMP 是相对寻址方式的双字节指令，其中 rel 为相对偏移量。指令的功能是按计算得到转移目的地址，实现程序转移。计算公式为

目的地址 =（PC）+2+rel

其中，PC 称为源地址，即指令“SJMP rel”所在程序单元的地址；偏移量 rel 是一个带符号的 8 位二进制补码数。如果 rel 为正，则向前转移；如果 rel 为负，则向后转移。计算偏移量的公式为

rel= 目的地址 -（源地址 +2）

若相对转移指令是 3 字节指令，则偏移量为

rel= 目的地址 -（源地址 +3）

例：　在 835AH 处有 SJMP 指令

835AH　　SJMP　　35H

源地址=835AH，rel=35H 且为正，则目的地址=835AH+02H+35H=8391H，即程序转移到8391H 地址。

例： 在 835AH 处的 SJMP 指令为

835AH SJMP 0E7H

rel=0E7H 且为负数 19H 的补码，因此目的地址=835AH+02H-19H=8343H，即程序转移到 8343H 处。

若 rel=FEH，为负数 02H 的补码，则目的地址=PC+02-02=PC，即目的地址和指令源地址相同，程序就在该指令上踏步，即

HERE：SJMP HERE 或 HERE：SJMP $

在 MCS-51 指令系统中，以$代表指令源地址。

若 rel=00H，则目的地址=PC+02H，即目的地址为下一条指令地址。如：

```
      SJMP 00H
NEXT：MOV  A，#00H
```

程序转移到 NEXT 处。

(4) 变址寻址转移指令

JMP @A+DPTR ;(PC)←(A)+(DPTR)

以 DPTR 内容为基础（称为基址)，A 中的内容作为变址。当 DPTR 固定时，A 中赋值不同，可以实现程序的对分支转移。其计算公式为

转移目的地址= (A)+(DPTR)

这种由基址寄存器（DPTR）和变址寄存器（A）共同实现的间址方式，称为变址寻址。

2. 条件转移指令

执行条件转移指令时，如指令中规定的条件满足，则进行程序转移；否则，程序顺利执行。

(1) 累加器判零转移指令

JZ rel ;若(A)=0，则(PC)←(PC)+2+rel，即转移
;否则(PC)←(PC)+2，即顺序执行

JNZ rel ;若(A)≠0，则(PC)←(PC)+2+rel，即转移
;否则(PC)←(PC)+2，即顺序执行

上述两条指令均为双字节指令。第一条指令转移条件是(A)=0，第二条指令转移条件是(A)≠0，A 中的内容为转移指令前面最后一条指令的执行结果。单片机的程序状态字 PSW 中没有零标志，只能用累加器的内容为零（非零）作为判断条件。

(2) 比较条件转移指令

比较条件转移指令是把两个操作数进行比较，以是否相等作为条件来控制程序转移。共有 4 条指令：

CJNE A，#data，rel ；累加器内容与立即数不等则转移，否则顺序执行

CJNE A，direct，rel ；累加器内容与内 RAM 中指定单元内容不等则转移，否则顺序执行

CJNE Rn，#data，rel ；工作寄存器内容与立即数不等则转移，否则顺序执行

CJNE @Ri，#data，rel ；内部 RAM 中指定单元（间址形式）内容与立即数不等则转移，否则顺序执行

上述 4 条指令是 3 字节指令，具有数值比较和程序转移两方面功能。

两个操作数比较结果影响 CY 标志，但不影响操作数中的内容。当左操作数=右操作数时，(CY)=0，程序顺序执行；当左操作数>右操作数时，(CY)=0，程序转移执行；当左操作数<右操作数时，(CY)=1，程序转移执行。

(3) 减 1 条件转移指令

这是一组把减 1 与条件转移两种功能结合在一起的指令，共有两条。

● 寄存器减 1 条件转移指令（双字节指令）为

DJNZ Rn，rel　；(Rn) ← (Rn) - 1

若 (Rn) ≠ 0，则 (PC) ← (PC) +2+rel，即程序转移；

若 (Rn) = 0，则 (PC) ← (PC) +2，即程序顺序执行。

● 直接寻址单元减 1 条件转移指令（3 字节指令）为

DJNZ direct，rel；(direct) ← (direct) -1

若 (direct) ≠ 0，则 (PC) ← (PC) + 3 + rel，即程序转移；

若 (direct) = 0，则 (PC) ← (PC) +3，即程序执行。

这两条指令主要用于控制程序循环。如预先把寄存器或内部 RAM 单元赋值循环次数，利用减 1 条件转移指令，以减 1 后是否为 0 作为转移条件，即可实现按次数控制循环。

例： 将外部 RAM 地址为 1100H～11FFH 的 256 个单元清 0，试编制实现程序。

```
       MOV   R7，#00H    ；置计数初值
       MOV   A，#00H
       MOV   DPTR，#1100H     ；清 0 单元首地址
LOOP：MOV    @DPTR，A         ；清 0
       INC   DPTR
       DJNZ  R7，LOOP         ；计数值减 1，不为 0 则循环
       RET                    ；返回
```

3．子程序调用及返回指令

从主程序转向子程序的指令称为子程序调用指令；从子程序返回主程序的指令称为返回指令。

调用指令与钻仪指令的主要区别是转移指令不保存返回地址，而子程序调用指令在转向目的地址的同时，必须保留返回地址（称为断点地址），以便执行返回指令时回到主程序断点的位置。通常采用堆栈技术保存断点地址,这样可以允许多重子程序调用（在子程序中再次调用子程序）。

（1 ）绝对调用指令（双字节指令）

ACALL addr11　；$(PC) \leftarrow (PC) + 2$，$(SP) \leftarrow (SP) + 1$，$(SP) \leftarrow (PC)_{7\sim0}$

　　　　　　　；$(SP) \leftarrow (SP) + 1$，$(SP) \leftarrow (PC)_{15\sim8}$

　　　　　　　；$(PC)_{10\sim0} \leftarrow addr11$，$(PC)_{15\sim11}$ 保留

该指令格式为

第一字节	A10	A9	A8	1	0	0	0	1
第二字节	A7	A6	A5	A4	A3	A2	A1	A0

指令代码中提供了子程序入口地址的低 11 位。这 11 位地址的 A7～A0 占据指令的第二字节，A10～A8 占据指令的第一字节的高 3 位，低 5 位为操作码。指令的调用范围为 2KB。

为了实现直程序调用，该指令共完成两项操作：

断点保护　断点保护是通过自动方式的堆栈操作实现的。即把加 2 以后的 PC 值（称为 PC 当前值）自动送入栈区保存起来，待子程序返回时再送回 PC。

构造目的地址　目的地址的构造是在 PC 加 2 的基础上，以提供的 11 位地址取代 PC 当前值中的低 11 位，PC 的高 5 位保持不变。

例： 在程序存储器 8100H 单元处有一条绝对调用指令，确定子程序目的地址。

8100H　　ACALL　　48FH

由于 48FH=0100 1000 1111B，即 addr11 的高 3 位（A10 A9 A8）=100，因此指令第一字节为 91H，第二字节为 8FH，即机器码为 918FH。

PC的当前值PC=8102H=1000 0001 0000 0010指令提供的低11位地址替换PC中的低11位后，形成的目的地址是

1000 0100 1000 1111B=848FH

即被调用的子程序入口地址为848FH。本指令的地址为8100H，不变的高5位是1000B，因此本指令的调用范围是8000H～87FFH(2 KB).

（2 ）长调用指令（3 字节指令）

ACALL addr16 ;（PC）←（PC）+3，（SP）←（SP）+1，（SP）←（PC）$_{7\sim0}$
;（SP）←（SP）+1，（SP）←（PC）$_{15\sim8}$
;（PC）←addr16

子程序入口地址在指令中直接给出.指令执行后，断点进栈保存,addr16作为子程序入口地址.本指令的调用范围是64KB(0000H～FFFH),使用比较方便,但3字节指令较ACALL指令占有较多的存储空间.

例18 已知下列程序段:

```
      ORG   0100H
      MOV   SP,#60H
      ……
      ORG   0200H
START：LCALL    MIR
      ……
      RET
      MIR  EQU  8100H
      END
```

程序执行结果：(SP) =62H，(61H) =03H，(62H) =02H，(PC) =8100H。

（3 ）返回指令

● 子程序返回指令

RET ;（PC）$_{15\sim8}$ ←（SP），（SP）←（SP）-1，（PC）$_{7\sim0}$ ←（SP），（SP）←（SP）-1

● 中断返回指令为

RETI ;（PC）$_{15\sim8}$ ←（SP），（SP）←（SP）-1，（PC）$_{7\sim0}$ ←（SP），（SP）←（SP）-1

子程序返回和中断返回指令的功能都是从堆栈中取出16位断点地址送PC，使子程序返回主程序。RET指令安排在子程序出口处，RETI指令安排在中断服务程序出口处。

此外，RETI指令还具有清除中断响应时被触发的优先级状态，开放较低级中断和恢复中断逻辑等功能。

例19 已知（SP）=62H，(62H）=07H，(61H）=30H，执行RET指令后，其结果是：(SP）=60H，(PC）=0730H，即CPU从0730H处开始执行程序。

4.空操作指令

NOP ;（PC）←（PC）+1

空操作指令也是一条控制指令，控制CPU不做任何操作，只消耗一个机器周期的时间。空操作指令是单字节指令，依次执行后PC加1，时间 延续一个机器周期。NOP指令常用于程序的等待或时间的延迟。

5 位操作类指令

位操作（又称位处理）就是以位（bit）为单位进行的运算和操作。位变量也称为布尔变量或开关变量。

MCS-51 指令系统适用位操作的地址空间是片内 RAM 20H～2FH 单元（位地址为 00H～7FH）以及 SFR 区中可寻址的位。

1.位传送指令

MOV C，bit ;（CY）←（bit）
MOV bit，C ;(bit) ← (CY)

bit 表示位地址。位传送就是可寻址的位与 CY 之间的相互传送。由于没有可寻址位之间的直接传送指令，因此位之间无法实现直接传送。如果需要位之间传送，必须以 CY 作中介实现。

例：将位地址为 20H 的内容传送到位地址 5AH。编制程序如下：

MOV 10H，C ;暂存 CY 内容
MOV C，20H ;20H 位送 CY
MOV 5AH，C ;CY 送 5AH 位
MOV C，10H ;恢复 CY 内容

2.位置位和复位指令

SETB C ;（CY）←1
SETB bit ;(bit) ←1
CLR C ;(CY) ←0
CLR bit ;(bit) ←0

3.位运算指令

位运算都是逻辑运算，有“与”、“或”、“非”3 种，共 6 条指令

ANL C，bit ;(CY) ← (CY) ∧ (bit)
ANL C，/bit ;(CY) ← (CY) ∧ ($\overline{bit}$)
ORL C，bit ;(CY) ← (CY) ∨ (bit)
ORL C，/bit ;(CY) ← (CY) ∨ ($\overline{bit}$)
CPL C ;(CY) ← ($\overline{CY}$)
CPL bit ;(CY) ← ($\overline{bit}$)

“/bit”表示位中内容的“非”，运算后 bit 中的内容不取反，保持原内容不变。

在位操作指令中，没有位的“异或”运算，需要时可由上述多条位操作指令实现。此外，通过位逻辑运算，可对各种组合逻辑电路进行模拟，即用软件方法来获得组合电路的逻辑功能。

例 21 用位运算指令实现“异或”操作：

$$D=E \oplus B$$

由于 D=E ⊕ B=EB+EB，实现的程序如下：

MOV C，B
ANL C，/E ;(CY) ←$\overline{E}$B
MOV D，C
MOV C，E
ANL C，/B ;(CY) ←E$\overline{B}$
ORL C，D ;$\overline{E}$B+E$\overline{B}$
MOV D，C ;D=E$\overline{B}$+$\overline{E}$B

4.位控制转移指令

位控制转移指令就是以位的状态作为实现程序转移的判断条件。

（1）以C 状态为条件的转移指令（双字节指令）

JC rel ；若（CY）=1，则（PC）←（PC）+2+rel，即转移

；若（CY）=0，则（PC）←（PC）+2，即程序顺序执行

JNC rel ；若（CY）=0，则（PC）←（PC）+2+rel，即转移

；若（CY）=1，则（PC）←（PC）+2，即程序顺序执行

（2）以bit 状态为条件的转移指令（3 字节指令）

JB bit，rel ；若（bit）=1，则（PC）←（PC）+3+rel，即转移

；若（bit）=0，则（PC）←（PC）+3，即程序顺序执行

JNB bit，rel ；若（bit）=0，则（PC）←（PC）+3+rel，即转移

；若（bit）=1，则（PC）←（PC）+3，即程序顺序执行

JBC bit，rel ；若（bit）=1，则（PC）←（PC）+3+rel，即转移，且同时伴随着清bit位，即（bit）=0

；若（bit）=0，则（PC）←（PC）+3，即程序顺序执行

JBC指令中，若可寻位为1时，则转移，并同时清该位。当bit是P0～P3端口中某一位时，该指令称为：“读-修改-写”指令。

4 汇编语言程序设计

用助记符表示的指令就是计算机的汇编语言，每一条指令就是汇编语言的一条语句。

所谓程序设计就是编写计算机程序。汇编语言程序设计就是使用汇编指令来编写计算机程序。

1 汇编语言的特点及其语句格式

1.汇编语言的特点

汇编语言有以下特点

1） 助记符指令与机器指令一一对应，所以用汇编语言编写的程序占用存储器空间小，运行速度快，可编写出最优化程序。

2）汇编语言是面向计算机的。汇编语言的程序设计人员必须对计算机硬件有相当深入的了解。

3） 汇编语言能直接访问存储器及接口电路，也能处理中断，因此汇编语言程序能直接管理和控制硬件设备。

4）各种计算机都有自己的汇编语言，不同计算机的汇编语言之间不能通用，因此汇编语言缺乏通用性，程序不易移植。

2.汇编语言的语句格式

各种计算机汇编语言的语句格式及语法规则基本相同。MCS-51汇编语言的语句格式为

[标号]:[操作码][目的操作数],[源操作数];[注释]

其中每部分也称为字段。各部分之间用一个空格或字段分界符分隔。常用的字段分界符有冒号“：”、逗号“，”和分号“；”。

（1）标号

标号用来说明指令的地址，用于其他语句对该句的访问。标号有以下规定：

1）标号由1～8个字母和数符组成，字母打头，冒号“：”结束，中间允许数字符号。标号中的字符个数不超过8个，若超过8个，则以前面的8个字符有效，后面的字符不起作用。

2）不能用本汇编语言已经定义的符号作为标号，如指令助记符、伪指令以及寄存器的符号名称符。

3）同一标号在一个程序中只能定义一次，不能重复定义。

4）一条语句可以有标号，也可以没有标号，取决于本程序中有无语句访问这条语句。

（2）操作码

操作码是汇编语句格式中惟一不能空缺的部分，用于规定语句执行的操作内容。

（3 ）操作数

操作数用于表明指令操作的数据或数据存储地址。操作数可以是空白，也可以是一项、两项，各操作数之间用逗号分开。MCS-51 指令系统的操作数有寄存器、立即数、直接、间接等7种寻址方式。

操作数与操作码之间用空格分开。

（4 ）注释

注释不属于语句的功能部分，只是对语句的解释说明，只要用“；”号开头，即表明以下为注释的内容。使用注释可使程序文件编制显得更加清楚，帮助程序人员阅读程序。注释可有可无，长度不限，一行不够时可以换行接着写，但换行时要注意在开头使用“；”号。

（5 ）分界符

分界符（分隔符）用于把语句格式中的各部分隔开，以便区分，包括空格、冒号、分号或逗号等多种符号。

冒号（：）——用于标号之后。

空格（ ）——用于操作码和操作数之间。

分号（；）——用于注释之前。

逗号（,）——用于操作数之间。

3.汇编语言程序设计的特点

汇编语言程序设计有以下特点：

1） 在程序中要对存取数据的存储器单元地址以及寄存器等作出明确分配。

2） 设计人员对单片机应用系统的硬件结构要有详细了解，以便在程序中熟练使用。

3） 设计程序要尽量采用模块化结构，便于阅读和修改。

4） 在满足工艺要求和便于阅读的基础上，尽量选用字节少，工作进行效率高的指令和结构形式。

2 汇编语言程序的基本结构形式

一般把程序结构分为3 种形式：顺序结构、分支结构和循环结构。

1.顺序结构

顺序结构是最简单的程序结构，在顺序程序中无分支、循环和调用子程序，程序是逐条顺序执行的。

例22 被加数存于片内RAM 32H，31H和30H；加数存于片内RAM 35H，34H和H；相加之和存于片内RAM 32H，31H和30H；进位存于00H单元，试编制程序。

```
START:  MOV     R0，#30H    ；被加数低字节地址
        MOV     R1，#33H    ；加数低字节地址
        MOV     A，@R0
        ADD     A，@R1      ；低字节相加
        MOV     @R0，A      ；存低字节相加结果
        INC     R0
        INC     R1
        MOV     A，@R0
        ADDC    A，@R1      ；中间字节相加
        MOV     @R0，A      ；存中间字节相加结果
```

```
        INC     R0
        INC     R1
        MOV     A, @R0
        ADDC    A, @R1          ；高字节相加
        MOV     @R0, A          ；存高字节相加结果
        CLR     A
        ADDC    A, #00H
        MOV     00H, A          ；存进位
        RET
```

2.分支结构

分支结构是通过转移指令实现的。根据程序的功能特点，又可分为单分支程序、多分支程序等。

例 23 假定在外 RAM 2000H，2001H，2002H 的 3 个连续单元中，2000H 和 2001H 单元存放着两个无符号数，要求找出其中较大者并存于 2002H 单元。其程序如下：

```
        ORG   0100H
START：  CLR   C
        MOV   DPTR，#2000H         ；设置数据指针
        MOVX  A，@DPTR             ；取第一个数
        MOV   R2，A                ；暂存于 R2
        INC   DPTR                 ；数据指针加 1
        MOVX  A，@DPTR             ；取第二个数
        SUBB  A，R2                ；两数比较
        JNC   LOOP1                ；第二个数大则转 LOOP1
        XCH   A，R2                ；第一个数大则交换
LOOP0：  INC   DPTR
        MOVX  @DPTR，A             ；存大数
        RET
LOOP1：  MOVX  A，@DPTR
        SJMP  LOOP0
```

3.循环结构

循环是为了重复执行一个程序段。在汇编语言中可以通过条件判断循环是否结束。

例 将内部 RAM 20H 为起始地址的数据串（最大长度为 32 字节）传送到外部 RAM 2000H 为首地址的区域，直到发现“$” 字符的 ASC II 码为止。其程序如下：

```
        MOV   R0，#20H        ；内 RAM 数据串首地址
        MOV   DPTR，#2000H    ；外 RAM 数据串首地址
        MOV   R7，#20H        ；最大数据串长度
LOOP0：  MOV   A，R0
        XRL   A，#24H         ；判断是否为“$”字符
        JZ    LOOP1
        MOV   A，@R0
        MOVX  @DPTR，A
```

```
          INC     R0
          INC     DPTR
          DJNZ    R7，LOOP0
LOOP1：   RET
```

5 汇编语言的伪指令与汇编

用指令系统编写的汇编语言程序称为源程序，必须将其翻译成机器码（称为目标程序），单片机方可执行。源程序转换成目标程序的过程是由通用计算机执行一种特定的翻译程序（称为汇编程序）自动完成的。这个翻译过程称为汇编。

1 汇编语言的伪指令

源程序中应有向汇编程序发出指示信息，告诉汇编程序如何完成汇编工作的控制命令，称之为伪指令。伪指令具有控制汇编程序的输入 / 输出、定义数据和符号、条件汇编和分配存储空间等功能。不同的汇编语言的伪指令也有所不同，但一些基本的东西却是相同的。

伪指令是由程序员发给汇编程序的命令，也称为汇编命令或汇编程序控制指令。只有在汇编前的源程序中才有伪指令，汇编后得到的目标程序（机器码）中没有伪指令相应的机器代码。

下面介绍 MCS-51 汇编语言程序中常见的伪指令。

1.ORG 汇编起始地址命令

在汇编语言源程序的开始，通常都要用一条 ORG（Origin）伪指令规定程序的起始地址。命令格式为

[标号]：ORG [地址]

其中：[标号]是选择项，根据需要选用；[地址]项通常为 16 绝对地址，但也可以使用标号或表达式。例如：

```
          ORG   8000H
START：   MOV   A，#00H
          ……
```

即规定标号 START 代表地址 8000H，目标程序的第一条指令从 8000H 开始。

2.END 汇编终止命令

END（END of assembly）是汇编语言源程序的结束标志，在整个源程序中只能有一条 END 命令，且位于程序的最后。如果 END 命令出现在中间，则其后面的源程序汇编时将不予处理。命令格式为

[标号]：END

命令中的[标号]是选择项。这个标号应是源程序第一条指令的符号地址。例如：

```
          ORG   8100H
START：   MOV   A，#00H
          MOV   R7，#10H
          MOV   R0，#20H
LOOP：    MOV   @R0，A
          INC   R0
          DJNZ  R7，LOOP
          RET
          END
```

3.EQU 赋值命令

EQU（Equate）命令用于给标号赋值。赋值以后，其符号值在整个程序中有效。命令格式为

[字符名称] EQU [赋值项]

其中,[赋值项]可以是常数、地址、标号或表达式。其值为 8 位或 16 位而进制数。赋值以后的字符名称既可以作立即数使用，也可以作地址使用。例如：

```
        ORG     6000H
START： MOV     R7，#05H
LOOP：  LCALL   DELAY
        DJNZ    R7，LOOP
        RET
        DELAY   EQU  1880H
        END
```

4.DB 定义字节命令

DB（Define Byte）命令用于从指定的地址开始，在程序存储器的连续单元中定义字节数据。命令格式为

[标号]:DB [8 位数据表]

字节数据可以是一字节常数或字符，或用逗号分开的字符串，或用引号括起来的字符串。例如：

DB“How are you？”

把字符串中的字符按 ASC II码存于连续的 ROM 单元中。

常使用本命令存放数据表格，例如存放数码管显示的十六进制数的形码，可使用多条 DB 命令定义：

```
DB  3FH，06H，5BH，4FH
DB  66H，6DH，7DH，07H
DB  7FH，6FH，77H，7CH
DB  0C0H，0F9H，0A4H，0B0H
```

5.DW 定义字命令

DW（Define Word）命令用于从指定地址开始，在程序存储器的连续单元中定义 16 位的数据字。命令格式为

[标号]:DW [16 位数据表]

存放时，数据的高 8 位在前（低地址)，低 8 位在后（高地址)。例如：

```
DW  “AA”              ；存入 41H，41H
DW  “A”               ；存入 00H，41H
DW  “ABC”             ；不合法，因超过两个字节
DW  100H，1ACH，814   ；按顺序存入 01H，00H，01H，ACH，FCH，DCH
```

DB 和 DW 定义的数据表，数的个数不得超过 80 个。如果数据的数目较多时，可使用多个定义命令。在 MCS-51 程序设计中，常以 DB 定义数据，以 DW 是定义地址。

6.DS 定义存储区命令

DS（Define Storage）命令用于从指定地址开始，保留指定树木的字节单元作为存储区，供程序运行使用。汇编时，这些单元不赋值。命令格式为

[标号]: DS [16 位数据表]

例如：

ADDTAL ： DS 20

从标号 ADDTBL 带表的地址开始，保留 20 个连续的地址单元。又例如：

ORG 8100H

DS 08H

从 8100H 地址开始，保留 8 个连续的地址单元。

注意 DB，DW 和 DS 命令只能对程序存储器使用，而不能对数据存储器使用。

6.BIT 位定义命令

本命令用于给字符名称赋以位地址。命令格式为

[字符名称] BIT [位地址]

其中[位地址]可以是绝对地址，也可以是符号地址（即位符号名称）。例如：

AQ BIT P1.0

把 P1.0 的位地址赋给变量 AQ。在其后的编程中，AQ 就可以作为位地址（P1.0）使用。

2 汇编语言的汇编

将用助记符编写的源程序转换成机器码的过程称为汇编。汇编分为手工汇编和机器汇编。

对于简单的应用程序，可以通过查表翻译指令的方法将源程序翻译成机器码，称之为手工汇编 。

由于手工汇编是按绝对地址进行定位，所以手工汇编时要根据转移的目标地址计算转移指令的偏移量，而且容易出错。此外，对于汇编后的目标程序，如须增加、删除和修改指令，就会引起以后各指令地址的改变，转移指令的偏移量也要重新计算。因此，手工汇编不是理想的方法，通常只用于小的程序。

编写完单片机的源程序之后，由于单片机本身软硬件资源所限，无法由单片机本身自动汇编（机器汇编），只能借助于通用计算机对源程序进行汇编。

使用一种计算机的汇编程序去汇编另一种计算机源程序，具体说就是运行汇编程序进行汇编的是一种计算机，而运行汇编得到目标程序的则是另一种计算机。这种使用一种计算机的汇编程序去汇编另一种计算机的源程序的汇编过程，被称为交叉汇编。单片机的机器汇编就是交叉汇编。

在交叉汇编之前，一般还要借助于通用计算机进行单片机的程序设计。通常使用编辑软件进行源程序的编辑，以形成一个由汇编指令和伪指令组成的源程序文件。这个过程被称为机器编辑。

交叉汇编之后，再使用串行通信方法，把汇编得到的目标程序传送到单片机，进行程序的调试和运行。

“机器编辑→交叉编辑→串行发送”，这 3 个过程构成了单片机软件设计的 3 个基本步骤。

源程序编写如下：

```
         ORG  8000H
START：  MOV  R0,  #20H
         MOV  R7,  #07H
         CLR  F0
LOOP：   MOV  A,   @R0
         MOV  2BH, A
         INC  R0
         MOV  2AH, @R0
         CLR  C
         SUBB A,   @R0
```

```
        JC    NEXT
        MOV   @R0, 2BH
        DEC   R0
        MOV   @R0, 2AH
        INC   R0
        SETB  F0
NEXT：  DJNZ  R7, LOOP
        JB    F0, START
HERE：  SJMP  $
        END
```

手工汇编结果如下表所列。

手工汇编结果

目标程序部分		源程序部分		
地址	机器码	标号	助记符指令	备注
8000	7820	START：	MOV R0，#20H	
8002	7F07		MOV R7，#07H	
8004	C2D5		CLR F0	
8006	E6	LOOP：	MOV A，@R0	
8007	F52B		MOV 2BH，A	
8009	08		INC R0	
800A	862A		MOV 2AH，@R0	
800C	C3		CLR C	
800D	96		SUBB A，@R0	
800E	4008		JC NEXT	偏移1
8010	A62B		MOV @R0，2BH	
8012	18		DEC R0	
8013	A62A		MOV @R0，2AH	
8015	08		INC R0	
8016	D2D5		SETB F0	
8018	DFEC	NEXT：	DJNZ R7，LOOP	偏移2
801A	20D5E3		JB F0，START	偏移3
801D	80FE	HERE：	SJMP $	偏移4

偏移 1 的计算：

rel1= 目的地址 -（源地址 +2）=8018H-（800EH+2）=08H

偏移 2 的计算：

rel2= 目的地址 -（源地址 +2）=8006H-（8018H+2）=-14H

(-14H）补码 =ECH

偏移3 的计算：

rel3=目的地址-（源地址+2）=8000H-（801AH+3）=-1DH

(-1DH）补码=E3H

偏移4 的计算：

rel4=目的地址-（源地址+2）=801DH-（801DH+2）=-2H

(-2H）补码=FEH

6 汇编语言程序设计举例

1 算术运算程序

1．加、减运算程序

（1）不带符号的多字节数加法

例 设有两个4字节的二进制数，分别存放在以30H和50H为起始地址的单元中（先存放低字节)。求这两个数的和，并将和存放在以30H 为起始地址的单元中，试编制程序。

程序如下：

```
            ORG   2000H
JAZ：       MOV       R0，#30H       ；指向加数最低位
            MOV       R1，#50H       ；另一加数最低位
            MOV       R2，#04H       ；字节个数存于R2
            LCALL     JAFA           ；调用加法子程序
            JC        OVER           ；有进位则转出
            MOV       34H，#00H      ；无进位清最低字节单元
            SJMP      HERE
OVER：      MOV       34H，#01H      ；最高字节单元为01H
HERE：      SJMP      HERE
            ORG       1000H
JAFA：      CLR       C              ；C清0
JAADD：     MOV       A，  @R0       ；取出加数一个字节
            ADDC      A，  @R1       ；加上另一个数的一个字节
            MOV       @R0，A         ；保存和
            INC       R0             ；修改加数的地址
            INC       R1
            DJNZ      R2，JAADD      ；没加完则继续
            RET
```

（2）不带符号的两个多字节数减法

例 设有两个N字节无符号数分别存于内RAM单元中，低字节在前，高字节在后。由R0指定被减数单元地址，由R1指定减数单元地址，要求差值存放在原被减数单元中，假定最高字节没有错位。

程序如下：

```
            CLR   C
            MOV   R7，#N          ；设定N字节
LOOP：      MOV   A，@R0          ；从低位取被减数字节
```

```
        SUBB    A，@R0          ；两位数减
        MOV     @R0，A          ；保存差
        INC     R0
        INC     R1
        DJNZ    R7，LOOP
        RET
```

（3 ）带符号数加、减运算

对于带符号数的减法运算，只要将减数的符号位取反，就可把减法运算按加法运算处理。

对于带符号数的加法运算，首先要进行两数符号的判定。如果两数符号相同，应进行两数相加，并以被加数符号为结果符号。

如果两数符号不同，应进行两数相减。如果相减的差为正，则差即为最后结果，并以被减数符号为结果符号；如果相减的差为负，则应将其差值取补，并把被减数的符号取反作为结果符号。

例27 假定20H和21H以及22H和23H分别存放两个16位的带符号二进制数，其中20H和22H的最高位为两数的符号位。请编写带符号双字节二进制数的加减法程序，以BUSB为减法程序入口，以BADD为加法程序入口，以内RAM 24H和25H保存运算结果。

程序如下：

```
BUSB：  MOV     A，22H          ；取减数高字节
        CPL     ACC.7
        MOV     22H，A          ；减数符号位取反进行加法
BADD：  MOV     A，20H          ；取被加数
        MOV     C，ACC.7
        MOV     F0，C           ；被加数符号位存于F0
        XRL     A，22H          ；两数高字节“异或”
        MOV     C，ACC.7        ；两数同号（CY）=0，异号（CY）=1
        MOV     A，20H          ；取被加数
        CPL     ACC.7           ；被加数高字节符号位清0
        MOV     20H，A          ；取其数值部分
        MOV     A，22H          ；取加数
        CLR     ACC.7           ；加数高字节符号位清0
        MOV     22H，A          ；取其数值部分
        JC      JIAN            ；两数异号转JIAN
JIA：   MOV     A，21H          ；两数同号进行加法
        ADD     A，23H          ；低字节相加
        MOV     25H，A          ；保存低字节和
        MOV     A，20H
        ADDC    A，22H          ；高字节相加
        MOV     24H，A          ；保存高字节和
        JB      ACC.7，QAZ ；符号位为1转溢出处理
QWE：   MOV     C，F0           ；结果符号处理
        MOV     ACC.7，C
        MOV     24H，A
        RET
```

```
JIAN：   MOV     A，21H          ；两数异号进行减法
         CLR     C
         SUBB    A，23H          ；低字节相减
         MOV     25H，A          ；保存差
         MOV     A，20H
         SUBB    A，22H          ；高字节相减
         MOV     24H，A          ；保存差
         JNB     ACC.7，QWE      ；没借位转QWE
BMP：MOV     A，25H          ；有借位，差值取补
         CPL     A
         ADD     A，#01H
         MOV     25H，A
         MOV     A，24H
         CPL     A
         ADDC    A，#00H
         MOV     24H，A
         CPL     F0              ；符号位取反
         SJMP    QWE
QAZ：……                     ；溢出处理（从省略）
```

2.乘法运算

对于单字节乘法运算，使用一条乘法指令MUL AB即可；对于多字节的乘法就必须通过程序实现。

例 假设被乘数存放于R6和R7中，乘数存放于R4和R5中，乘积存放于40H，41H，42H和43H中，低字节在前，双字节乘法结果最多为4字节。

双字节乘法按一般竖式相乘原理，设R6×R4=H64，L64；R7×R4=H74，L74；R5×R6=H56，L56；R7×R5=H75，L75。其中，H表示高字，L表示低字节。竖式乘法过程表示为

```
                R7      R6
       ×)       R5      R4
       ---------------------------
                        H64     L64  ←—  R6×R4
                H74     L74  ←———————  R7×R4
                H56     L56  ←———————  R5×R6
        H75     L75  ←———————————————  R7×R5
       ---------------------------
       (43H)   (42H)   (41H)   (40H)
```

具体程序如下:

```
         ORG     0020H
MUL16:   MOV     R0,#40H         ；积地址指针
         MOV     A，R6
         MOV     B，R4
         MUL     AB              ；R6×R4=H64，L64
         MOV     @R0，A          ；L64→（40H）
         MOV     R3，B       ；H64→R3
         MOV     A，R7
```

```
            MOV     B , R 4
            MUL     AB            ; R7 × R4=H74, L74
            ADD     A , R 3       ; L74+H64 → R3
            MOV     R 3 , A
            MOV     A, #0 0 H     ; H74+CY → R2
            MOV     R 2 , A
            MOV     A , R 6
            MOV     B , R 5
            MUL     AB            ; R5 × R6=H56, L56
            ADD     A, R3      ; L56+L74+H64 → A
            INC     R0
            MOV     @R0, A        ; A → (41H)
            MOV     R1, #00H
            MOV     A, R2
            ADDC    A, B          ; H56+R2+CY → R2
            MOV     R2, A
            JNC     NEXT
            INC     R1
NEXT：      MOV     A, R7
            MOV     B, R5
            MUL     AB            ; R7 × R5=H75, L75
            ADD     A, R2      ; L75+R2 → A
            INC     R0
            MOV     @R0, A        ; A → (42H)
            MOV     A, B
            ADDC    A, R1      ; H75+R1+CY → A
            INC     R0
            MOV     @R0, A
            RET
```

3.除法运算

对于单字节除法运算使用一条除法指令 DIV AB 即可；但对于多字节的除法就必须通过程序实现。

多字节除法的程序设计常采用“恢复余数法”，其设计思想是做减法。

仿照手工算法进行除法，设被除数为 100011，除数为 101，求 100011B ÷ 101B= ？

	0 0 0 1 1 1	
除数	1 0 0 0 1 1	……………被除数
	-) 1 0 1	……………2^{-1} ×除数
	1 1 1	……………余数
	-) 1 0 1	……………2^{-2} ×余数
	1 0 1	……………余数
	-) 1 0 1	……………2^{-3} ×除数
	0	

计算机除法运算采用“左移被除数相除法”。做除法前先将余数单元清0，在CY=0条件下，执行左循环移位，将被除数最高位移入余数单元最低位，被除数最低位变为0，然后用余数减去除数。若够减，则此时被除数移位单元最低位置1，即商为1，同时用差取代余数；若不够减，则此时的被除数移位单元仍为0，即商为0。这样重复移位，做减法，直到被除数全部左移入余数单元。最后被除数移位单元变成了商数单元，余数单元存有余数。

设被除数为1011，除数为0101，余数单元全清0，下面是采用左移位除法过程。

第一次移位：余数单元=0001，被除数移位单元=0110，余数单元减去除数，不够减，继续左移。

第二次移位：余数单元=0010，被除数移位单元=1100，余数单元减去除数，不够减，继续左移。

第三次移位：余数单元=0101，被除数移位单元=1000，余数单元减去除数，够减且差为0000，用此时的差值取代原来余数，并将被除数移位单元最低位置1，即余数单元=0000，被除数移位单元=1001，继续左移。

第四次移位：余数单元=0001，被除数移位单元=0010，移位完成，最后结果是：商为0010，余数为0001。

例 编写一个16位÷16位除法程序。假设被除数存于40H和41H中，除数存于44H和45H中，商存于40H和41H中，余数存于42H和43H中。低字节在前，48H和49H为暂存单元。

程序如下：

```
          ORG     0059H
DIV16:    MOV     R0,#40H         ；被除数为0则退出
          MOV     A，@R0
          JNZ     LOP0
          INC     R0
          MOV     A，@R0
          JNZ     LOP0
          CLR     A
          MOV     42H，A
          MOV     43H，A
          RET
LOP0：    MOV     R0,#44H         ；除数为0则退出
          MOV     A，@R0
          JNZ     LOP1
          INC     R0
          MOV     A，@R0
          JNZ     LOP1
          RET
LOP1：    CLR     A
          MOV     42H，A          ；清余数单元42H和43H
          MOV     43H，A
          MOV     R2,#10H         ；置移位次数
LOP2：    CLR     C               ；CY=0
          MOV     R3，#04H
```

```
        MOV     R0，#40H        ；被除数地址指针
LOP3：  MOV     A，@R0          ；余数单元，被除数单元左移一次
        RLC     A
        MOV     @R0，A
        INC     R0
        DJNZ    R3，LOP3
        MOV     R0，#42H        ；余数单元减除数
        MOV     R1，#44H
        MOV     A，@R0
        CLR     C
        SUBB    A，@R1
        MOV     48H，A          ；暂存差的低字节
        INC     R0
        INC     R1
        MOV     A，@R0
        SUBB    A，@R1
        MOV     49H，A          ；暂存差的高字节
        JC      LOP4            ；不够减继续左移
        MOV     R0，#42H        ；够减时差值取代原余数
        MOV     R1，#48H
        MOV     A，@R1
        MOV     @R0，A
        INC     R0
        INC     R1
        MOV     A，@R1
        MOV     @R0，A
        MOV     A，40H
        INC     A               ；够减时被除数单元加1
        MOV     40H，A
LOP4：  DJNZ    R2，LOP2        ；移位次数不到，继续
        RET
        END
```

2 数制转换程序

1.十六进制数转换成ASCⅡ码

例 在片内RAM 20H单元中存有2位十六进制数，将其转换成ASCⅡ码，并存于21H和22H两个单元中。

程序如下：

```
        MOV     SP，#3FH
MAIN：  PUSH    20H             ；十六进制数进栈
        LCALL   HASC            ；调用转换子程序
        POP     21H             ；第一位转换结果送21H单元
        MOV     A，20H          ；再取原十六进制数
```

```
        SWAP    A               ；高低半字节交换
        PUSH    ACC             ；交换后的十六进制数进栈
        LCALL   HASC            ；调用转换子程序
        POP     22H             ；第二位转换结果送22H单元
        RET
HASC：  DEC     SP              ；跨过断点保护对象
        DEC     SP
        POP     ACC             ；弹出转换数据
        ANL     A，#0FH         ；屏蔽高4位
        ADD     A，#07H         ；修改变址寄存器内容
        MOVC    A，@A+PC        ；查表
        PUSH    ACC             ；查表结果进栈
        INC     SP              ；修改堆栈指针回到断点保护内容
        INC     SP
        RET
ASCTAB： D B“0，1，2，3，4，5，6，7”  ；ASCⅡ码表
        D B“8，9，A，B，C，D，E，F”
```

2.ASCⅡ码转换成十六进制数

例 将外部RAM 30H～3FH单元中的ASCⅡ码依次转换为十六进制数，并存入内RAM 60H～67H单元中。

程序如下：

```
MAIN：  MOV     R0，#30H        ；设置ASCⅡ码地址指针
        MOV     R1，#60H        ；设置十六进制数地址指针
        MOV     R7，#08H        ；需拼装的十六进制数的字节数
LOOPA： LCALL   TRAN            ；调用转换子程序
        SWAP    A               ；A中高低4位交换
        MOV     @R1，A          ；存于内部RAM
        INC     R0
        LCALL   TRAN            ；调用转换子程序
        XCHD    A，@R1          ；十六进制数拼装
        INC     R0
        INC     R1
        DJNZ    R7，LOOPA
        RET
TRAN：  CLR     C
        MOVX    A，@R0          ；取ASCⅡ码
        SUBB    A，#30H         ；减去30H
        CJNE    A，#0AH，LOOPB
        SJMP    LOOPC
LOOPB： JC      DONE
LOOPC： SUBB    A，#07H
DONE：  RET
```

3 定时程序

在单片机应用系统中，定时功能除可使用定时器 / 计数器实现外，还可使用定时程序完成。定时程序是典型的循环程序，是通过执行一个具有固定延迟时间的循环体来实现延时的。

1.单循环定时程序

```
        MOV     R7，#TIME
LOOP：  NOP
        NOP
        DJNZ    R7，LOOP
        RET
```

NOP 指令的机器周期为 1，DJNZ 指令的机器周期为 2，则一次循环共 4 个机器周期。如果单片机的晶振频率为 6MHz，则一个机器周期是 2 μs，因此一次循环的延迟时间为 8 μs。定时程序的总延迟时间是循环程序段的整数倍，该程序的延迟时间为 8 × TIME（μs)。这个程序的最长延时时间为 256 × 8=2048 μs。

2.较长时间的定时程序

为了加长定时时间，通常采用多重循环的方法。如下面的双重循环的定时程序，最长可延时 262 914 个机器周期，即 525 828 μs 或大约 526ms（晶振频率为 6MHz)。

```
        MOV     R7，#TIME1      ；1 个机器周期
LOOP1： MOV     R6，#TIME2      ；1 个机器周期
        NOP                     ；1 个机器周期
        NOP                     ；1 个机器周期
        DJNZ    R6，LOOP2       ；2 个机器周期
        DJNZ    R7，LOOP1       ；2 个机器周期
        RET                     ；2 个机器周期
```

最长定时时间计算公式为

$$(256 \times 4+2+1) \times 256 \times 2+4=525\ 828\ \mu s$$

3.以一个基本的延时程序满足不同的定时要求

如果系统中有多个定时需要，可以先设计一个基本的延时程序，使其延迟时间为各定时时间的最大公约数，然后以此基本程序作为子程序，通过调用的方法实现所需要的不同定时。例如：要求的定时时间分别为 5s，10s 和 20s，设计一个 1s 延时子程序 DELAY，则不同定时的调用情况表示如下（晶振频率为 6MHz)：

```
        MOV     R5，#05H        ；延时 5s
LOOP1： LCALL   DELAY
        DJNZ    R5，LOOP1
        RET
        MOV     R5，#0AH        ；延时 10s
LOOP2： LCALL   DELAY
        DJNZ    R5，LOOP2
        RET
        MOV     R5，#14H        ；延时 20s
LOOP3： LCALL   DELAY
        DJNZ    R5，LOOP3
```

```
        RET
DELAY：  MOV     R7，#0FAH
LOOPA：  MOV     R6，#0FAH
LOOPB：  NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        DJNZ    R6，LOOPB
        DJNZ    R7，LOOPA
        RET
```

延时时间为

$$(250\times 8+2+1)\times 250\times 2+4=1\ 001\ 504\ \mu s\approx 1s$$

4.查表程序

预先把数据形式存放在程序存储器中，然后使用程序读出。这种能读出表格数据的程序被称为查表程序。MCS-51 指令系统准备了专用的查表指令：

MOVC A，@A+DPTR

MOVC A，@A+PC

这两个 MOVC 指令的功能是完全相同的。它们在不改变 DPTR 和 PC 的状态下，只根据 A 的内容就可以取出表格中的数据。但这两条指令在具体使用上也存在差异。前一条指令的基址寄存器 DPTR 能提供 16 位基址，而且还能在使用前给 DPTR 赋值，查表空间可达 64KB。后一条指令是以 PC 作为基址寄存器，虽然也能提供 16 位地址，但 PC 不能被赋值，所以其基址值是固定的。由于 A 的内容为 8 位无符号数，因次只能在当前指令下面的 256 个地址单元内进行查表，即数据只能放在该指令后面的 256 个地址单元之内，而且表格只能被程序段所使用。

例　设有一个巡回检测报警装置，需要对 16 路输入值进行比较，当每一路输入值等于或超过该路的报警值时，实现报警。下面根据这一要求，编制一个查表程序。

设 Xi 为路数，查表是 Xi 按 0，1，2，…，15（i=15）取数，表中报警值是 2 字节数，依 Xi 顺序列成表格放在 TAB 中。进入查表程序之前，路数 Xi 放在 R2 中，其输入值存于 R0 和 R1 当中，查表结果若许报警，将 P1.0 置 1，否则清 0。

```
        ORG     1000H
TB1:    MOV     A,R2            ；路数 Xi → R2 → A
        ADD     A,R2            ；R2+R2 → A
        MOV     R2,A            ；A → R2
        MOV     DPTR,#TAB       ；取数据表首地址
        MOVC    A，@A+DPTR ；取出高字节
        MOV     R4,A            ；高字节→ R4
        INC     R2              ；地址指向低字节
        MOV     A,R2
        MOVC    A，@A+DPTR ；取出低字节
        MOV     R3,A            ；低字节→ R3
        CLR     C
```

```
        MOV     A,R0            ；当前输入值与报警值比较
        SUBB    A, R3       ；低字节相减
        MOV     A,R1
        SUBB    A, R4       ；高字节相减
        JNC     LOOP
        CLR     P1.0            ；输入值＜报警值
        RET                     ；返回
LOOP：  SETB    P1.0            ；输入值≥报警值
        RET                     ；返回
        ORG     2000H
TAB：DW     05F0H，0E89H，0A69H，1EAAH
        DW      0D9BH，7F93H，0373H，26D7H
        DW      2710H，9E3FH，1A66H，22E3H
        DW      1174H，16EFH，33E4H，6CA0H
        END
```

5　数据极值查找程序

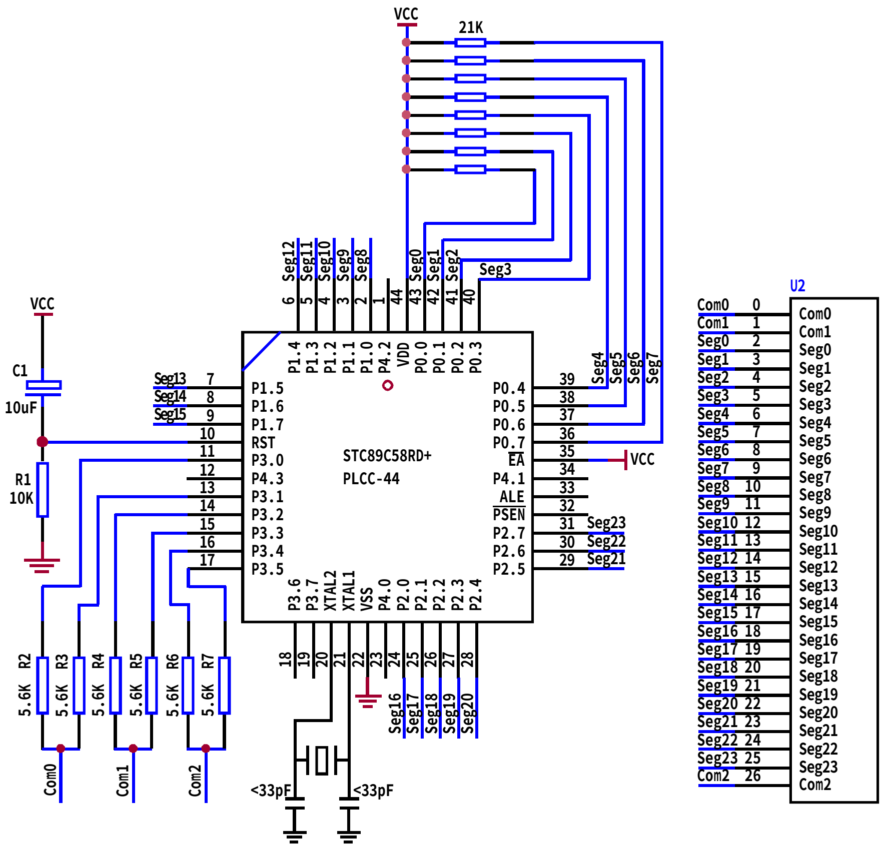
极值查找就是在指定的数据区中挑出最大值或最小值。

例　片内RAM　20H 单元开始存放 8 个无符号 8 位二进制数，找出其中的最大值。极值查找操作的主要内容是进行数值大小的比较。假定在比较过程中，以 A 存放大数，与之逐个比较的另一个数放在 3AH 单元中。比较结束后，把查找到的最大数送到 3BH 单元中。

程序如下：

```
        MOV     R0，#20H        ；数据区首地址
        MOV     R7，#08H        ；数据区长度
        MOV     A，@R0          ；读第一个数
LOOP：  INC     R0
        MOV     3AH，@R0        ；读下一个数
        CJNE    A，3AH，CHK     ；数值比较
        SJMP    LOOP1
CHK：   JNC     LOOP1           ；A 值大则转
        MOV     A，@R0          ；大数送 A
LOOP1： DJNZ    R7，LOOP        ；继续比较
        MOV     3BH，A
        RET
```

附录L：　STC 单片机普通 I/O 口驱动 LCD 显示



本资料不提供技术支持，请自行消化吸收

```
NAME LcdDriver
$include(STC89C51RC.h)

;*************************************************************************************
;the LCD is 1/3 duty and 1/3 bias; 3Com*24Seg; 9 display RAM;
;
;                      Bit7    Bit6   Bit5   Bit4   Bit3   Bit2   Bit1   Bit0
;Com0:  Com0Data0:    Seg7    Seg6   Seg5   Seg4   Seg3   Seg2   Seg1   Seg0
;       Com0Data1:    Seg15   Seg14  Seg13  Seg12  Seg11  Seg10  Seg9   Seg8
;       Com0Data2:    Seg23   Seg22  Seg21  Seg20  Seg19  Seg18  Seg17  Seg16
;Com1:  Com1Data0:    Seg7    Seg6   Seg5   Seg4   Seg3   Seg2   Seg1   Seg0
;       Com1Data1:    Seg15   Seg14  Seg13  Seg12  Seg11  Seg10  Seg9   Seg8
;       Com1Data2:    Seg23   Seg22  Seg21  Seg20  Seg19  Seg18  Seg17  Seg16
;Com2:  Com2Data0:    Seg7    Seg6   Seg5   Seg4   Seg3   Seg2   Seg1   Seg0
;       Com2Data1:    Seg15   Seg14  Seg13  Seg12  Seg11  Seg10  Seg9   Seg8
;       Com2Data2:    Seg23   Seg22  Seg21  Seg20  Seg19  Seg18  Seg17  Seg16
;*************************************************************************************
;Com0:  P3^0,P3^1   when P3^0 = P3^1 = 1       then Com0=VCC(=5V);
;                        P3^0 = P3^1 = 0       then Com0=GND(=0V);
;                        P3^0 = 1, P3^1=0      then Com0=1/2 VCC;
;Com1:  P3^2,P3^3   the same as the Com0
;Com2:  P3^4,P3^5   the same as the Com0
;

sbit SEG0  =P0^0
sbit SEG1  =P0^1
sbit SEG2  =P0^2
sbit SEG3  =P0^3
sbit SEG4  =P0^4
sbit SEG5  =P0^5
sbit SEG6  =P0^6
sbit SEG7  =P0^7
sbit SEG8  =P1^0
sbit SEG9  =P1^1
sbit SEG10 =P1^2
sbit SEG11 =P1^3
sbit SEG12 =P1^4
sbit SEG13 =P1^5
sbit SEG14 =P1^6
sbit SEG15 =P1^7
sbit SEG16 =P2^0
sbit SEG17 =P2^1
sbit SEG18 =P2^2
sbit SEG19 =P2^3
```

```
sbit SEG20 =P2^4
sbit SEG21 =P2^5
sbit SEG22 =P2^6
sbit SEG23 =P2^7
;*****************************************************************************************

;======Interrupt===============================
       CSEG AT 0000H
       LJMP start

       CSEG AT 000BH
       LJMP int_t0

;======register===============================
lcdd_bit SEGMENT BIT
       RSEG lcdd_bit
       OutFlag:       DBIT 1          ;the output display reverse flag

lcdd_data SEGMENT DATA
      RSEG lcdd_data
      Com0Data0:     DS   1
      Com0Data1:     DS   1
      Com0Data2:     DS   1
      Com1Data0:     DS   1
      Com1Data1:     DS   1
      Com1Data2:     DS   1
      Com2Data0:     DS   1
      Com2Data1:     DS   1
      Com2Data2:     DS   1
      TimeS:         DS   1

;======Interrupt Code=========================
t0_int SEGMENT CODE
     RSEG  t0_int
     USING 1
;******************************************************************
;Time0 interrupt
;ths system crystalloid is 22.1184MHz
;the time to get the Time0 interrupr is 2.5mS
;the whole duty is 2.5mS*6=15mS, including reverse
;******************************************************************
int_t0:
    ORL  TL0,#00H
    MOV  TH0,#0EEH
    PUSH ACC
    PUSH PSW
```

```
    MOV   PSW,#08H
    ACALL OutData
    POP   PSW
    POP   ACC
    RETI

;======SUB CODE================================
uart_sub SEGMENT CODE
        RSEG  uart_sub
        USING 0
;*******************************************************************
;initial the display RAM data
;if want to display other,then you may add other data to this RAM
;Com0:   Com0Data0,Com0Data1,Com0Data2
;Com1:   Com1Data0,Com1Data1,Com1Data2
;Com2:   Com2Data0,Com0Data1,Com0Data2
;********************************************************************
InitComData:                        ;it will display "11111111"
    MOV Com0Data0,#24H
    MOV Com0Data1,#49H
    MOV Com0Data2,#92H
    MOV Com1Data0,#92H
    MOV Com1Data1,#24H
    MOV Com1Data2,#49H
    MOV Com2Data0,#00H
    MOV Com2Data1,#00H
    MOV Com2Data2,#00H
    RET

;*********************************************************************
;reverse the display data
;*********************************************************************
RetComData:
    MOV  R0,#Com0Data0              ;get the first data address
    MOV  R7,#9
RetCom_0:
    MOV  A,@R0
    CPL  A
    MOV  @R0,A
    INC  R0
    DJNZ R7,RetCom_0
    RET
```

```
;*************************************************************************
;get the display Data and send to Output register
;*************************************************************************
OutData:
    INC   TimeS
    MOV   A,TimeS
    MOV   P3,#11010101B              ;clear display,all Com are 1/2VCC and invalidate
    CJNE A,#01H,OutData_1           ;judge the duty
    MOV   P0,Com0Data0
    MOV   P1,Com0Data1
    MOV   P2,Com0Data2
    JNB   OutFlag,OutData_00
    MOV   P3,#11010111B              ;Com0 is work and is VCC
    RET
OutData_00:
    MOV   P3,#11010100B              ;Com0 is work and is GND
    RET
OutData_1:
    CJNE A,#02H,OutData_2
    MOV   P0,Com1Data0
    MOV   P1,Com1Data1
    MOV   P2,Com1Data2
    JNB   OutFlag,OutData_10
    MOV   P3,#11011101B              ;Com1 is work and is VCC
    RET
OutData_10:
    MOV   P3,#11010001B              ;Com1 is work and is GND
    RET
OutData_2:
    MOV   P0,Com2Data0
    MOV   P1,Com2Data1
    MOV   P2,Com2Data2
    JNB   OutFlag,OutData_20
    MOV   P3,#11110101B              ;Com2 is work and is VCC
    SJMP OutData_21
OutData_20:
    MOV   P3,#11000101B              ;Com2 is work and is GND
OutData_21:
    MOV    TimeS,#00H
    ACALL RetComData
    CPL    OutFlag
    RET
```

```
;======Main Code================================
uart_main SEGMENT CODE
        RSEG  uart_main
        USING 0

start:
           MOV   SP,#40H
           CLR   OutFlag
           MOV   TimeS,#00H
           MOV   TL0,#00H
           MOV   TH0,#0EEH
           MOV   TMOD,#01H
           MOV   IE,#82H
           ACALL InitComData
           SETB  TR0
Main:
     NOP
     SJMP Main

END
```

STC89C51RC/RD+系列单片机准双向口输出原理

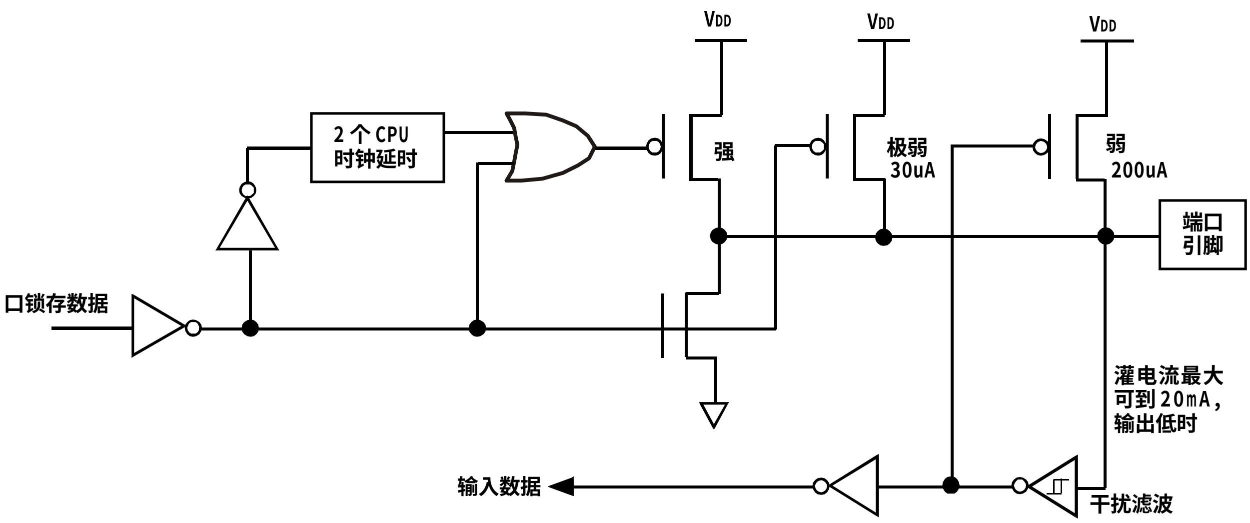
准双向口输出类型可用作输出和输入功能而不需重新配置口线输出状态。这是因为当口线输出为 1 时驱动能力很弱，允许外部装置将其拉低。当引脚输出为低时，它的驱动能力很强，可吸收相当大的电流。准双向口有 3 个上拉晶体管适应不同的需要。

在 3 个上拉晶体管中，有 1 个上拉晶体管称为 " 弱上拉 "，当口线寄存器为 1 且引脚本身也为 1 时打开。此上拉提供基本驱动电流使准双向口输出为 1。如果一个引脚输出为 1 而由外部装置下拉到低时，弱上拉关闭而 " 极弱上拉 " 维持开状态，为了把这个引脚强拉为低，外部装置必须有足够的灌电流能力使引脚上的电压降到门槛电压以下。

第 2 个上拉晶体管，称为 " 极弱上拉 "，当口线锁存为 1 时打开。当引脚悬空时，这个极弱的上拉源产生很弱的上拉电流将引脚上拉为高电平。

第 3 个上拉晶体管称为 " 强上拉 "。当口线锁存器由 0 到 1 跳变时，这个上拉用来加快准双向口由逻辑 0 到逻辑 1 转换。当发生这种情况时，强上拉打开约 2 个机器周期以使引脚能够迅速地上拉到高电平。

准双向口输出如下图所示。



STC89LE51RC/RD+ 系列单片机为 3V 器件，如果用户在引脚加上 5V 电压，将会有电流从引脚流向 VDD，这样导致额外的功率消耗。因此，建议不要在准双向口模式中向 3V 单片机引脚施加 5V 电压,如使用的话，要加限流电阻，或用二极管做输入隔离，或用三极管做输出隔离。

准双向口带有一个施密特触发输入以及一个干扰抑制电路。

附录 M：STC89 系列单片机和 STC12 系列单片机双 CPU 通信

--- 主机用软件实现 SPI 通信，从机用硬件实现 SPI 通信

```
;/* ---  STC International Limited ---------------------------------------------- */
;/* ---  宏晶科技      姚永平     2006/7/1     V1.0 ------------------------------- */
;/* ---  one_master_one_slave --------------------------------------------------- */
;/* ---  STC89C51RC,    STC89C52RC,    STC89C53RC     做主机用普通 I/O 口软件实现 SPI  - */
;/* ---  STC89C54RD+,   STC89C58RD+,   STC89C516RD+   做主机用普通 I/O 口软件实现 SPI  - */
;/* ---  STC12C5412AD, STC12C5410AD, STC12C5408AD  做从机用自身的硬件 SPI -------- */
;/* ---  STC12C5052AD, STC12C4052AD, STC12C2052AD  做从机用自身的硬件 SPI -------- */
;/* ---  Mobile: 13922805190 --------------------------------------------------- */
;/* ---  Fax: 0755-82944243 ---------------------------------------------------- */
;/* ---  Tel: 0755-82948409 ---------------------------------------------------- */
;/* ---  Web: www.MCU-Memory.com ----------------------------------------------- */
;/* ---  如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 -------- */
;/* ---  如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序 -------- */

;-------------------------------------------------------------------------------
;1. 本示例程序演示 STC89C/LE5xx 系列 MCU 用任意 I/O 口实现 SPI 功能。本程序适用
;    于单主单从系统。
;-------------------------------------------------------------------------------
;2. 硬件连接:三线连接
;
;            单片机                                          从 SPI 器件
;
;       +--------------+  MISO     <-- 位流方向     MISO +--------------+
;       |   任意 I/O 口 |<<----------------------------|      SPI       |
;       |              |                              |8 位移位寄存器  |
;       |   任意 I/O 口 |---------------------------->>|                |
;       +-------+------+  MOSI     位流方向 -->     MOSI +-------^------+
;               |                                                |
;               |          SCLK                     SCLK         |
;               +--------------------->>---------------------+
;
;     除此之外，单片机的 RS-232 串行口通过 RS-232 转换器与 PC 机的 RS-232
;串行口相连接。
;-------------------------------------------------------------------------------
;3. SPI 通讯过程：
;3.1 通讯原理:
;     单片机每向"从 SPI 器件"(或从单片机)的 SCLK 脚发送一个脉冲，就有 1 bit
;单片机数据由 MOSI 脚传送到"从 SPI 器件"的 8 位移位寄存器中。与此同时有 1 bit
;"从 SPI 器件"数据由 MISO 脚传送到单片机中。也可以认为单片机和"从 SPI 器件"
;互换了 1 bit 数据。
;3.2 单片机发送 1 字节到"从 SPI 器件"
;     单片机的发送子程序每次将 8 bit 数据发送(由 8 个 SCLK 脉冲驱动) 到从
;SPI 器件的 8 位移位寄存器中。与此同时有 8 bit 数据由"从 SPI 器件"传回单片机,
;但单片机的发送子程序未予处理这 8 bit 数据。
```

```
;3.3 单片机接收"从 SPI 器件"的 1 字节
;      单片机的接收子程序每次向 SCLK 脚发送 8 个脉冲，将"从 SPI 器件"传回的
; 8位移位寄存器数据保存起来备用。与此同时有 8 bit 未知数据由单片机传送到了
;"从 SPI 器件"的 8位移位寄存器中，在实际应用中请注意对此进行处理。
;4.  SPI 工作模式
;      由三个特性决定 SPI 工作模式
;    a) 数据传输顺序: 先传低位 LSB 还是先传高位 MSB, 在本例中由 SOFT_DORD 设定
;    b) 时钟信号极性: 空闲时 SCLK 脚的电平, 即 SCLK 的 "前跳变沿" 是下降沿还是
;                      上升沿。在本例中由 SOFT_CPOL 设定时钟信号极性。
;    c) 时钟信号相位: 第一个跳变沿发生在第一个 bit 传输周期的 0 度 还是 180 度。
;                     即数据由  SCLK 的 "前跳变沿" 还是 "后跳变沿" 驱动到 SPI
;                      口线。在本例中由 SOFT_CPHA 设定时钟信号相位。
;5. 使用方法
;    a) 修改程序，使 SPI 工作模式与 "从 SPI 器件" 的工作模式相同。
;    b) 将汇编后的程序代码下载到从单片机中。
;    c) 给电路板上电。
;    d) 用串口调试助手(STC 的 ISP 下载程序 STC-ISP.exe 3.2 以上版本提供了该功能)
;       向单片机发送一串数据。
;          单片机的 RS-232 串口每收到一个字节就立刻将收到的字节通过 SPI 口
;       发送到 "从 SPI 器件" 中。接下来接收 "从 SPI 器件" 的一个字节，并把这个
;       字节通过 RS-232 口发送到 PC 机显示出来。
;6. 从 SPI 器件
;    可以用 STC12C2052AD 和 STC12C5410AD 系列 MCU 作为 "从 SPI 器件" 调试本程序。
;7. 注意事项:
;       "从 SPI 器件" 的速度较慢时, 要在 SPI 口接收, 发送子程序中插入空操作指令
;    NOP。
;8. 实验结果: 使用 STC12C5410AD, STC89C5xx 系列 MCU, SPI 口传输数据无误。
;---------------------------------------------------------------------------
;定义常量
;-----------------
;定义波特率自动重装数常量
;以下波特率是 PCON.7 = 0 时的数值, 若使 PCON.7 = 1 可将波特率加倍
RELOAD_8BIT_DATA  EQU 0FFH    ;Fosc=22.1184MHz, Baud = 57600
;RELOAD_8BIT_DATA  EQU 0FBH   ;Fosc=18.432MHz, Baud=9600, 1T 运行时 Baud=115200
;RELOAD_8BIT_DATA   EQU 0F6H  ;Fosc=18.432MHz, Baud=4800, 1T 运行时 Baud=57600
;RELOAD_8BIT_DATA  EQU 0F3H   ;Fosc=12.0000, Baud = 2403
;RELOAD_8BIT_DATA  EQU 0FFH   ;Fosc=11.059MHz, Baud = 28800
;---------------------------------------------------------------------------
;定义特殊功能寄存器
AUXR    EQU 8EH
;---------------------------------------------------------------------------
;定义 SPI 脚
;SCLK     EQU P1.7
;MISO     EQU P1.6
;MOSI     EQU P1.5
SCLK    EQU P1.0
MISO    EQU P1.1
MOSI    EQU P3.0
```

```
;---------------------------------------------------------------------------------
;定义单片机管脚
LED_MCU_START       EQU P3.4
;---------------------------------------------------------------------------------
;定义 SPI 模式:
SOFT_DORD    EQU     1    ;SPI 数据传输顺序，1:先传低位 LSB，0:先传高位 MSB
SOFT_CPOL    EQU     1    ;SPI 时钟信号极性选择位
          ;1：SPI 空闲时 SPICLK = 1，前跳变沿是下降沿，后跳变沿是上升沿。
          ;0：SPI 空闲时 SPICLK = 0，前跳变沿是上升沿，后跳变沿是下降沿。
SOFT_CPHA    EQU     1    ;SPI 时钟信号相位选择位
          ;1：数据由 SPICLK 前跳变沿驱动到 SPI 口线，SPI 模块在后跳变沿采样数据。
          ;0：SPICLK 后跳变沿数据被改变(被驱动到口线)，在 SPICLK 前跳变沿数据被采样。
;---------------------------------------------------------------------------------
SET_SCLK_IDEL_VAL    MACRO            ;设置 SPI clock 时钟线空闲时电平
    if SOFT_CPOL
        SETB   SCLK                 ;1:SPI空闲时SPICLK=1,前跳变沿是下降沿,后跳变沿是上升沿
    else
        CLR    SCLK                 ;0:SPI空闲时SPICLK=0,前跳变沿是上升沿,后跳变沿是下降沿
    endif
ENDM

SET_SCLK_FOREPART_VAL   MACRO         ;设置 SCLK 脚 1 bit 周期前半程电平
    if SOFT_CPHA = 0
        if SOFT_CPOL = 0
            CLR    SCLK               ;SOFT_CPHA, SOFT_CPOL = 0,0
        else
            SETB   SCLK               ;SOFT_CPHA, SOFT_CPOL = 0,1
        endif
    else
        if SOFT_CPOL = 0
            SETB   SCLK               ;SOFT_CPHA, SOFT_CPOL = 1,0
        else
            CLR    SCLK               ;SOFT_CPHA, SOFT_CPOL = 1,1
        endif
    endif
ENDM

SET_SCLK_SECOND_HALF_VAL    MACRO     ;设置 SCLK 脚 1 bit 周期后半程电平
    if SOFT_CPHA = 0
        if SOFT_CPOL = 0
            SETB   SCLK               ;SOFT_CPHA, SOFT_CPOL = 0,0
        else
            CLR    SCLK               ;SOFT_CPHA, SOFT_CPOL = 0,1
        endif
    else
        if SOFT_CPOL = 0
            CLR    SCLK               ;SOFT_CPHA, SOFT_CPOL = 1,0
```

```
        else
            SETB  SCLK              ;SOFT_CPHA, SOFT_CPOL = 1,1
        endif
    endif
ENDM

OUTPUT_BIT_SPI_MOSI   MACRO         ;输出一 bit 到 MOSI 口线
    if SOFT_DORD
        RRC  A                      ;1:先传低位 LSB
    else
        RLC  A                      ;0:先传高位 MSB
    endif
    MOV   MOSI, C
ENDM

INPUT_BIT_SPI_MISO    MACRO         ;从 MISO 口线输入 1 bit
    MOV   C, MISO
    if SOFT_DORD
        RRC  A                      ;1:先传低位 LSB
    else
        RLC  A                      ;0:先传高位 MSB
    endif
ENDM
;-----------------------------------------------------------------------------
;定义变量
SPI_buffer       EQU     30H        ;SPI 字节收发缓冲区
;-----------------------------------------------------------------------------
    ORG   0000H
    LJMP  MAIN
;-----------------------------------------------------------------------------
    ORG   0080H
MAIN:
    CLR   LED_MCU_START             ;点亮 MCU 开始工作指示灯
    MOV   SP, #7FH
    ACALL Init_System               ;系统初始化

Check_RS232:
    JNB   RI, Check_RS232           ;判 RS-232 串口中收到数据否
    ;单片机 RS-232 串口已收到新的数据
    ACALL Get_Byte_From_RS232       ;单片机将RS-232串口中收到的数据送到累加器 A

;   ACALL RS232_Send_Byte           ;调试用，将累加器 A 中的数据发送到 PC 机

    ACALL SPI_Send_Byte             ;单片机将累加器 A 中的数据发送到"从 SPI 器件"
    ACALL Delay                     ;延时，等待"从 SPI 器件"准备好发送数据
    ACALL SPI_Receive_Byte          ;接收"从 SPI 器件"一个字节
    ;单片机 SPI 端口已收到新的数据
```

```
    MOV    A, SPI_buffer              ;将 "从 SPI 器件" 发回的数据"送到累加器 A
    ACALL RS232_Send_Byte             ;将累加器 A 中的数据发送到 PC 机
    SJMP   Check_RS232
;-------------------------------------------------------------------------------
Init_System:
    ACALL Initial_UART                ;初始化串口
    ACALL Initial_SPI                 ;初始化 SPI
    SETB   EA                         ;开总中断
    RET
;-------------------------------------------------------------------------------
Initial_UART:                         ;初始化串口
; SCON  Bit:    7        6      5      4      3      2      1     0
;             SM0/FE   SM1    SM2    REN    TB8    RB8     TI    RI
    MOV    SCON, #50H                 ;0101,0000 8位可变波特率，无奇偶校验

    MOV    TMOD, #21H                 ;T1 为自动重装模式
    MOV    TH1, #RELOAD_8BIT_DATA
    MOV    TL1, #RELOAD_8BIT_DATA

;     MOV    PCON, #80H                 ;取消本行指令注释，波特率加倍。

     ;使以下两行有效，波特率快 12 倍
    MOV    A, #01000000B               ;T1 以 1T 的速度计数，是普通 8051 的 12 倍
    ORL    AUXR, A

    SETB   TR1                        ;启动定时器 1 开始计数
    RET
;-------------------------------------------------------------------------------
Initial_SPI:                          ;初始化 SPI
    SET_SCLK_IDEL_VAL                 ;设置 SPI clock 时钟线空闲时电平
    SETB   MISO
    SETB   MOSI
    RET
;-------------------------------------------------------------------------------
RS232_Send_Byte:                      ;RS232 串口发送一个字节
    CLR    TI                         ;清零串口发送中断标志
    MOV    SBUF, A
RS232_Send_Wait:
    JNB    TI, RS232_Send_Wait        ;等待发送完毕，未发送完毕跳回本行
    CLR    TI                         ;清零串口发送中断标志
    RET
;-------------------------------------------------------------------------------
Get_Byte_From_RS232:                  ;取 RS-232 串口中收到的数据送累加器 A
    MOV    A, SBUF
    CLR    RI
    RET
```

```
;------------------------------------------------------------------------------
;将 A 中的数据用 SPI 口发送出去
SPI_Send_Byte:                          ;SPI 发送一个字节
    SET_SCLK_IDEL_VAL                   ;设置 SPI clock 时钟线空闲时电平
    MOV    R2, #8
SPI_Send_Loop:
    OUTPUT_BIT_SPI_MOSI                 ;输出一 bit 到 MOSI 口线
    SET_SCLK_FOREPART_VAL               ;设置 SCLK 脚 1 bit 周期前半程电平
;   NOP                                 ;为降低 SPI 通讯速率可插入数个 NOP
;   NOP
;   NOP
    SET_SCLK_SECOND_HALF_VAL            ;设置 SCLK 脚 1 bit 周期后半程电平
;   NOP                                 ;为降低 SPI 通讯速率可插入数个 NOP
;   NOP
    DJNZ   R2, SPI_Send_Loop
    SET_SCLK_IDEL_VAL                   ;设置 SPI clock 时钟线空闲时电平
    RET
;------------------------------------------------------------------------------
;用 SPI 接收一个字节送 A 和 SPI_buffer 中
SPI_Receive_Byte:                       ;SPI 接收一个字节
    SET_SCLK_IDEL_VAL                   ;设置 SPI clock 时钟线空闲时电平
    MOV    R2, #8
SPI_Receive_Loop:
    SET_SCLK_FOREPART_VAL               ;设置 SCLK 脚 1 bit 周期前半程电平
    NOP                                 ;等待 MISO 口线数据稳定，这个 NOP 是必需的
    NOP                                 ;对于高速的 1T 单片机，这个 NOP 是必需的
;   NOP                                 ;为降低 SPI 通讯速率可插入数个 NOP
    INPUT_BIT_SPI_MISO                  ;从 MISO 口线输入 1 bit
    SET_SCLK_SECOND_HALF_VAL            ;设置 SCLK 脚 1 bit 周期后半程电平
;   NOP                                 ;为降低 SPI 通讯速率可插入数个 NOP
;   NOP
    DJNZ   R2, SPI_Receive_Loop
    SET_SCLK_IDEL_VAL                   ;设置 SPI clock 时钟线空闲时电平
    MOV    SPI_buffer, A
    RET
;------------------------------------------------------------------------------
Delay:
    MOV    R2, #50H
Delay_Loop:
    DJNZ   R2, Delay_Loop
    RET
;------------------------------------------------------------------------------
    END
;------------------------------------------------------------------------------
```

附录N： 超强抗干扰测试数据，过4000V快速脉冲干扰

--- 在权威的SGS电气实验室测试

Iss Date Jan 06, 2006

SGS

WORKSHEET： EFT Test (EN 61000-4-4)

EUT NAME：STC89C51RC/RD+ Development Tool

Environmental conditions

Air Temperature (°C)	20	Relative Humidity (%RH)	56	Air Pressure (mBar)	1003

Set-Up EUT for Compliance Test

Test Requirement Standard	EN 61000-4-4	Set up the EUT as Table-top
Operating mode during test	ON MODE	
Detail EUT monitor system: STC12C5410AD control LED flashing function		

Compliance-Test Record

Compliance Test Result On AC Cable (2 wire)			
Lead under test	Level	EUT operating mode	Observations
L	+/-4kV	ON MODE	NLOF
N	+/-4kV	ON MODE	NLOF
L+N	+/-4kV	ON MODE	NLOF

Test Conclusion and Reason

Test Conclusion: Pass

Reason(explain why?): Class A

Note: This sample pass the Class A(2kV)

According to the customer's require up to 4kV,test result:PASS 4kV

我们提供过4000V快速脉冲干扰辅导服务，陪同在权威的SGS电气实验室测试，1小时收取人民币3000元，提前一周预约。

附录O: 资料升级历史备忘录

2006-3-3 版本在 2006-2-6 版本的基础上：

附录 I 关于 STC12C5410AD 系列 PWM/PCA 的应用部分增加了在使用 PCA 高速输出模式时的特别应用注意事项

2006-2-6 版本在 2005-1-16 版本的基础上：

增加了定时器 1 做波特率发生器的程序
增加了如何用软件实现系统复位
增加了附录 M 超强抗干扰测试数据，过 4000V 快速脉冲干扰
对 STC12C5410AD 系列单片机的 EEPROM 测试程序的解释说明部分做了加强
对 STC12C5410AD 系列单片机的看门狗测试程序的解释说明部分做了加强

2006-1-16 版本在 2005-12-31 版本的基础上：

1. 附录 I 关于 STC12C5410AD 系列 A/D 转换程序做了简化
2. 附录 I 关于 STC12C5410AD 系列 PCA/PWM 模块增加了新的演示程序（扩展软件定时器，扩展外部中断）

2005-12-31 版本在 2005-12-24 版本的基础上：

1. 修正了附录 I 关于 STC12C5410AD 系列单片机中 PCA/PWM 部分笔误，4 路 PCA/PWM 原有些部分笔误为 2 路
2. 附录 I 关于 STC12C5410AD 系列原 A/D 转换结果计算公式 ：

笔误 结果(ADC_DATA[7:0],ADC_LOW2[1:0]) = 256 x Vin / Vcc

现改为 结果 (ADC_DATA[7:0],ADC_LOW2[1:0]) = 1024 x Vin / Vcc

3. 增加了 STC 单片机普通 I/O 口驱动 LCD 显示的参考电路及演示程序